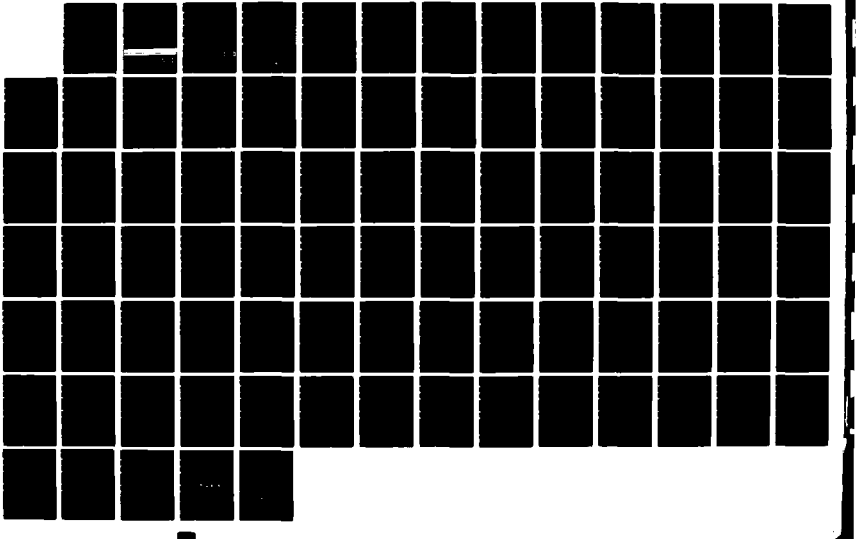
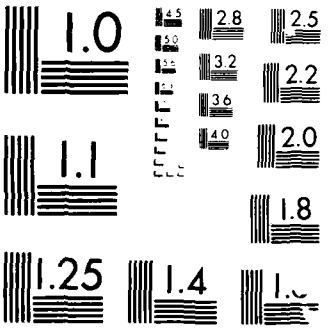
AD-A191 668 THEORETICAL INVESTIGATION OF OPTICAL COMPUTING BASED ON 1/1
NEURAL NETWORK MO (U) CALIFORNIA INST OF TECH PASADENA
DEPT OF ELECTRICAL ENGINEERI Y ABU-MOSTAFA ET AL
UNCLASSIFIED 29 SEP 87 AFOSR-TR-88-0025 AFOSR-86-0296 F/G 23/3 NL





(2)

AFOSR-TR- 88-0025

THEORETICAL INVESTIGATION OF OPTICAL COMPUTING BASED ON NEURAL NETWORK MODELS

Yaser Abu-Mostafa and Demetri Psaltis

CALIFORNIA INSTITUTE OF TECHNOLOGY

PASADENA, CALIFORNIA

Annual Report

THEORETICAL INVESTIGATION OF OPTICAL COMPUTING BASED ON NEURAL NETWORK MODELS

Yaser Abu-Mostafa and Demetri Psaltis

Grant AFOSR-86-0296

Submitted to:
Dr. Lee Giles
Air Force Office of Scientific Research
Bolling Air Force Base
Washington, DC

Principal Investigators

Demetri Psaltis and Yaser Abu-Mostafa
Department of Electrical Engineering
California Institute of Technology
Pasadena, CA 91125

SECURITY CLASSIFICATION OF THIS PAGE

AD-A191 668

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release, distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 88-0025
6a. NAME OF PERFORMING ORGANIZATION California Inst of Tech 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFOSR/NE
6c. ADDRESS (City, State, and ZIP Code) Dept of Electrical Engineering Pasadena, CA 91125	7b. ADDRESS (City, State, and ZIP Code) Bldg 410 Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Same as 7a 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-86-0296
8c. ADDRESS (City, State, and ZIP Code) Same as 7b	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61102F; PROJECT NO. 2305; TASK NO B1; WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
THEORETICAL INVESTIGATION OF OPTICAL COMPUTING BASED ON NEURAL NETWORK MODELS

12. PERSONAL AUTHOR(S)
Professor Demetri Psaltis

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Annual Report	FROM 30 Sep 86 TO 29 Sep 87		

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

It is difficult to find good mathematical models for many natural problems such as pattern recognition. Not only does this difficulty preclude finding good solutions for these problems, but it also precludes estimating their complexity using the standard tools of the theory of computational complexity (Traub, 1985). Part of the difficulty can be traced to symptoms such as ill-definition, fuzziness, and inexactness. However, the difficulty of modeling these problems may be inherent in some cases.

UNCLASSIFIED

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	
22a. NAME OF RESPONSIBLE INDIVIDUAL DR GILES	22b. TELEPHONE (Include Area Code) 202 767-4933
	22c. OFFICE SYMBOL NE

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

1 Random Problems

A problem (a Boolean function $f : \{0,1\}^N \mapsto \{0,1\}$) is characterized by its randomness (*à la Kolmogorov*) $R(f)$ and its entropy (*à la Shannon*) $H(f)$. Random problems have large values of $R(f)$, and are a good model for many natural pattern recognition problems. $R(f)$ and $H(f)$ are shown to be lower and upper bounds, respectively, for a minimum-size circuit that computes f. False entropy, namely the hidden structure of a problem, is related to the difference between $H(f)$ and $R(f)$.

1.1 Introduction

It is difficult to find good mathematical models for many natural problems such as pattern recognition. Not only does this difficulty preclude finding good solutions for these problems, but it also precludes estimating their complexity using the standard tools of the theory of computational complexity (Traub, 1985). Part of the difficulty can be traced to symptoms such as ill-definition, fuzziness, and inexactness. However, the difficulty of modeling these problems may be inherent in some cases. To illustrate what we mean, consider the following problem:

A. *Input:* 255-45-5237 *Output:* Is this the social security number of a convict?

To solve this problem in general, one needs a list of the social security numbers of all convicts. It is highly improbable that there exists a simple relation between the social security number and the legal status of a person. Barring such a relation, one cannot hope for an algorithm to 'manipulate' the input so as to arrive at the output. In other words, the above list cannot be compacted into a simple algorithm. Contrast this problem with the following familiar problem:

B. *Input:* 255,455,237 *Output:* Is this number a prime?

Although one can resort to consulting a list of primes, there is the option of writing a simple algorithm to test for primality and applying it to this number. It may take a long time to execute, but the algorithm itself is

short in comparison with the list of primes. What is the basic difference between problems A and B? The notion of a prime has a short effective definition, while the notion of a convict does not. The long list of social security numbers of convicts is the effective definition of the notion of a convict.

Problems which do not have a concise effective definition are called *random problems* (Abu-Mostafa, 1985). Randomness here is based on the length of the shortest algorithm (Kolmogorov, 1965), and has nothing to do with probability or fuzziness. If this length exceeds a certain threshold, the problem is considered random. On the other hand, for *structured problems* such as B, the algorithm can be quite short. The difficulty in modeling random problems is inherent. This is because an effective model could be viewed as an algorithm (not necessarily a very efficient one), and hence has to be long in the case of a random problem.

Many natural pattern recognition problems can be considered random problems. This fact is usually overlooked due to the apparent 'structure' some of these problems have, e.g., visual images which have many clear regularities. However, after these regularities are considered, a major random component is left. A complete model for visual scenes, or an algorithm for computer vision, will cover the random as well as the structured components of the problem, and hence will have to be sufficiently long.

Three factors contribute to the significance of random problems and widen their scope. First of all, the definition of algorithmic randomness is based on universal Turing machines (Turing, 1936) which are more powerful than any physical system. This makes some problems which are not truly random *look* random for all practical purposes and will have to be treated as such. The second factor is that there is no constructive way in general to find the shortest algorithm for an arbitrary problem (Kolmogorov, 1965). In spite of its generality, this fact reflects the difficulty of modeling and suggests that some problems will have be treated as random problems just because no one will be able to find their concise model. Finally, although it may be logically impossible to tell whether a problem is random (Chaitin, 1982), a problem generated by probabilistic means is very likely to be random. Therefore, the assumption that a natural problem is random is probably valid.

This last remark leads to another observation. Most of the practical random problems turn out to be ill-defined as well. However, we maintain separation of concerns in this paper. Only randomness, as defined above, is assumed in the problems we address here. Our aim is to characterize random problems and their computational demands.

In section 2, we introduce the formal definitions of randomness and entropy and prove some basic facts about them. In section 3, we relate these quantities to the complexity of implementing the function. Finally, we draw some insight into the class of random problems in section 4. We shall restrict ourselves to binary alphabets for simplicity; the generalization to arbitrary finite alphabets is straightforward. All logarithms and exponentials are to the base 2.

1.2 Randomness and entropy

Let N be an arbitrary fixed positive integer and consider the Boolean functions of the form $f : \{0,1\}^N \mapsto \{0,1\}$. Any such function is fully characterized by its truth table which can be listed as a (2^N-bit long) binary string $\tau(f) = \tau_0\tau_1 \cdots \tau_k \cdots \tau_{2^N-1}$, where τ_k is the value of f when the argument is the N-bit binary representation of the number k.

Let U denote a fixed universal Turing machine with binary input alphabet $\{0,1\}$. U takes a binary string $\mathbf{p}$ as input (program) and runs on $\mathbf{p}$ to produce an output string $\mathbf{s}$ (if it eventually halts). In this case, we say that $U(\mathbf{p}) = \mathbf{s}$. Based on U, the Kolmogorov complexity of a string $\mathbf{s}$ is defined by

$$K(\mathbf{s}) = \min \{|\mathbf{p}| \mid U(\mathbf{p}) = \mathbf{s}\},$$

where $|\mathbf{p}|$ denotes the length of the string $\mathbf{p}$.

The Kolmogorov complexity measures the degree of randomness of a string; if two binary strings of the same length have different Kolmogorov complexities, the one with the higher complexity is more 'random'. The randomness of a Boolean function f is based on the Kolmogorov complexity of its truth table;

$$R(f) = \log K(\tau(f)) \quad \text{bits},$$

where the logarithm (to the base 2) is taken to make the range of $R(f)$ run from ≈ 0 (where $\mathbf{p}$ is a very short program, hence $\tau(f)$ has a very regular

pattern) to $\approx N$ (where p is as long as $\tau(f)$, hence $\tau(f)$ has no pattern whatsoever).

A problem will be considered random if the corresponding function f has a large value of $R(f)$. How large? We fix a threshold R_0 and make the definition relative to R_0. The choice of a particular R_0 is not critical to most of the theory, and may therefore be motivated by practical considerations.

Definition. A problem $f : \{0,1\}^N \mapsto \{0,1\}$ is said to be *random* if $R(f) \geq R_0$.

Since $R(f)$ can be at most $\approx N$, the threshold R_0 should be smaller than N. This is necessary to make the definition interesting, i.e., to guarantee that some of the problems are indeed random. In fact, the overwhelming majority of all problems will be random even if R_0 is only a few bits smaller than N. To see this, we observe that there are 2^{2^N} problems (Boolean functions of N variables), while there are at most $2^0+2^1+\cdots+2^K < 2^{K+1}$ programs p of length $\leq K$. Therefore, At most $2^{2^{R_0}+1}$ problems can be *non-random*, and this is only a negligible fraction of 2^{2^N}.

On the other hand, if R_0 is not very small, it will be impossible to pinpoint a specific problem and prove that it is random. This is a consequence of Chaitin's version of Gödel's incompleteness theorem; there is a number K_0 (depends on the axiomatic system) such that no statement of the form '$K(\mathrm{s}) \geq K_0$' is provable (within the system). If we pick $R_0 \geq \log K_0$, where K_0 corresponds to axiomatic set theory, it will be impossible to prove (using regular mathematics) that any given problem is random.

The difficulty of proving randomness for specific problems is by no means a serious drawback for the notion of random problems. There are many cases where the probability that the problem is random is sufficiently high to warrant treating it as a random problem, in spite of the lack of a *proof* of randomness. In fact, whenever probability is involved in generating f, the chances are f will be a random problem.

Example. Let $f : \{0,1\}^N \mapsto \{0,1\}$, where N is large, be generated as follows. Each bit of the truth table $\tau(f) = \tau_0 \cdots \tau_{2^N-1}$ is (independently) set to 1 with probability ϵ and to 0 with probability $1-\epsilon$, where $0 < \epsilon < 1$. The expected number of 1's in $\tau(f)$ is therefore $\epsilon 2^N$. With high probability (law of large numbers), f will have about that many 1's in its truth table.

Therefore, f will be any of $\approx \binom{2^N}{\epsilon 2^N}$ functions with approximately equal probability. This number can be estimated as $\binom{2^N}{\epsilon 2^N} \approx 2^{\mathcal{H}(\epsilon)2^N}$, where $\mathcal{H}(\epsilon) = -\epsilon \log \epsilon - (1-\epsilon)\log(1-\epsilon)$. We can again enumerate the programs $\mathbf{p}$ of length 0, 1, etc., and conclude that $R(f) \geq N - \Delta$ with high probability, where Δ is only a few bits more than $-\log \mathcal{H}(\epsilon)$.

This example also illustrates that the randomness of a problem is affected by the number of 1's in the truth table. If the number of 1's is very small, one can write a short program $\mathbf{p}$ to generate $\tau(f)$ by specifying *where the 1's are* in $\tau(f)$. The same can be done in the dual case where the number of 0's is small. Problems which have few 1's (or few 0's) in their truth tables are of special interest because they model the cases where only a small fraction of inputs to f are of interest, a condition commonly encountered in natural problems. The quantity that captures this property is entropy.

Let $h(f) = \min\{\, |f^{-1}(1)|\,,\ |f^{-1}(0)|\,\}$, i.e., the number of 1's or the number of 0's (whichever is smaller) in the truth table of f. The (deterministic) entropy of f is defined by

$$H(f) = \log(1 + h(f)) \quad \text{bits.}$$

The logarithm (and the added 1) make the range of $H(f)$ run from 0 (the two constant functions) to $\approx N$ (functions with as many 1's as 0's). Hence, $H(f)$ has essentially the same range as $R(f)$.

Except for a small 'error' term (at most the order of $\log N$), $H(f)$ serves as an upper bound for $R(f)$. To see this, we assume that $\tau(f)$ has only a limited number of 1's (the dual case of 0's is similar) and write a relatively short program $\mathbf{p}$ that generates $\tau(f)$. The program is a listing of the locations of the 1's in $\tau(f)$. Since each location in $\tau(f)$ can be specified by N bits, the length of $\mathbf{p}$ is approximately $N2^{H(f)}$. The logarithm of that, which is an upper bound for $R(f)$, is $H(f) + \log N$. An enumeration of all programs of length 0, 1, etc., and of the number of functions of a certain level of entropy shows that $H(f)$ is a *tight* upper bound for $R(f)$. In fact, for most functions, $R(f) \approx H(f)$.

It is interesting to notice that $R(f)$ and $H(f)$ can be considered two extremes in a spectrum of quantities that measure the complexity of spec-

ifying f based on models of varying degree of sophistication. On the one hand, $H(f)$ mesures the complexity of specifying f if the specification is done by simply listing the 1's or the 0's of the function. Hence, the model on which the measure $H(f)$ is based is the 'lookup' model. On the other hand, the model on which $R(f)$ is based is the universal Turing machine. $R(f)$ measures the complexity of specifying f based on a very powerful tool that generates $\tau(f)$ from the specification. Hence $R(f)$ is as small as can be; it can take advantage of any effective property f may have. There are many models that fall between these two extremes. For example, a time-bounded version of $R(f)$ can be based on the time-bounded Kolmogorov complexity of $\tau(f)$. The underlying model will be a universal Turing machine that is allowed to run for only a limited number of steps. Another model which is treated in more detail in section 3 is combinational circuits, where regularities of f that can be captured by logic elements help reduce the size of the specification of f.

The difference between $H(f)$ and $R(f)$ is a significant quantity. It is called *false entropy* (Abu-Mostafa, 1986) and expresses how much 'hidden' structure the problem has. This interpretation follows from the above discussion; the model on which $H(f)$ is based is the lookup model that does not take any advantage of the location of the 1's and 0's of f, just their number, while the model on which $R(f)$ is based takes advantage of any effective regularity, no matter how subtle, to reduce the size of the specification. The difference expresses how much of the entropy of f can be removed if the structure of f is taken into consideration. The problem of pattern recognition hinges on removing as much of the false entropy as possible.

1.3 Complexity bounds

Consider the problem of implementing the function f, e.g., using a combinational circuit (Savage, 1976). We wish to estimate the complexity of such implementation, and investigate the relation between complexity, entropy, and randomness. Since N is fixed, the number of input instances is finite and our measure of complexity will be a nonuniform one, i.e., not based on a finite process that works for any of an infinite number of input instances.

Nonuniformity of the complexity measure in our context is more realistic for two reasons. First, the pattern recognition problems we are modeling are finite in nature with no clear extension into an infinite problem. Secondly, the systems that are projected for pattern recognition are based on learning, i.e., automatic development of the system from training samples. As such, the final system does not have to be uniform although the learning mechanism itself may be uniform.

There are several ways of defining circuits all of which are equivalent, and a corresponding number of ways of defining circuit complexity (size) all of which are within a constant (independent of N and f) from one another. For concreteness, we give one such definition in detail. Our circuits are combinational (loop-free), with unlimited fan-out (an output of a gate can be used as an input to any number of gates). The only type of gate we use is the two-input NAND gate (whose output is 0 if, and only if, both inputs are 1's) which by itself is a complete basis (any Boolean function can be simulated using a circuit consisting exclusively of copies of this gate). The independent Boolean variables (inputs of f) are called $x_1, \cdots, x_N$, and are available to be used as inputs to any gate in the circuit.

A circuit is a chain $\Gamma = \gamma_1 \cdots \gamma_Q$ of Q gates. The outputs of these gates are called $y_1, \cdots, y_Q$, respectively. Each gate γ_q can have as inputs any of the independent variables $x_1, \cdots, x_N$ as well as the outputs of the *previous* gates $y_1, \cdots, y_{q-1}$. Since all the gates are two-input NAND gates, we only need to specify the inputs to each γ_q. Therefore, formally, each γ_q is a pair (not necessarily distinct, order doesn't matter) of elements from the set $\{x_1, \cdots, x_N, y_1, \cdots, y_{q-1}\}$. Finally, the output of the circuit is the output of the last gate, y_Q. We say that a circuit Γ simulates a function f if $f = y_Q$ for all assignments of the variables $x_1, \cdots, x_N$. The number of gates in Γ, namely Q, is denoted by $c(\Gamma)$. The (circuit) complexity of a problem f is defined by

$$C(f) = \log \min\{c(\Gamma) \mid \Gamma \text{ simulates } f\} \quad \text{bits.}$$

Again we have the range of $C(f)$ running from ≈ 0 (simple functions) to $\approx N$ (complex functions). The fact that the maximum value $C(f)$ can assume is $\approx N$ follows from the exhaustive implementation of any Boolean function of N variables that uses $\approx 2^N$ gates (which can be improved to $2^N/N$ due to a classical result of Shannon). We now show that, except for

an error term of at most the order of $\log N$, the value $C(f)$ is at least $R(f)$ and at most $H(f)$, which we write as

$$R(f) \preceq C(f) \preceq H(f).$$

This relationship reflects the fact that combinational logic is more sophisticated than table lookup, but less sophisticated than universal Turing machines.

To see that $C(f) \preceq H(f)$, consider the case where f has $h(f)$ 1's (the dual case is similar). One can build a circuit with N one-input NOT gates, $h(f)$ N-input AND gates, and one $h(f)$-input OR gate to simulate f (implementing the 1's of the function directly by a sum of products). One can replace all these gates by at most $\alpha N(1 + h(f))$ two-input NAND gates (where α is a suitable constant). Hence $C(f) \leq H(f) + \log N + \text{constant}$. Therefore, apart from the error term, the bound is valid.

To see that $C(f) \succeq R(f)$, consider a program p that encodes a minimum-size circuit $\Gamma = \gamma_1 \cdots \gamma_Q$ that simulates f. Each γ_q is a pair of variables selected from at most $N + Q$ variables. Hence it takes at most $2\log(N+Q)$ bits to encode each γ_q. Hence, the program that encodes the whole circuit Γ will be at most $\alpha Q \log(N + Q)$ bits long (where α is a suitable constant). The logarithm of that is an upper bound for $R(f)$. By definition, $\log Q = C(f)$. The other (error) terms are at most the order of $\log N$ since Q is at most the order of 2^N.

The fact that $R(f)$ is a lower bound for $C(f)$ implies that random problems cannot be solved by small circuits.

1.4 Conclusion

The notion of random problems was introduced to capture the inherent difficulty of natural pattern recognition problems and estimate their computational demands. The paper introduced the main concepts and proved some basic facts for the idealized case where the problem is defined as a deterministic Boolean function. The results are summarized in the relationship

$$R(f) \preceq C(f) \preceq H(f).$$

The theory of rate-distortion may be employed to accommodate the practical case of continuous-valued functions. To accommodate the case where there is a probability distribution over the input variables, one can define a probabilistic version of the measures $R(f)$, $H(f)$, $C(f)$, e.g.,

$$R_\delta(f) = \min\{R(g) \mid \Pr(f \neq g) \leq \delta\}.$$

In words, the randomness of f with tolerance for error δ of the time is the minimum randomness of any function g that differs from f at most δ of the time.

We also made the remark that the difference between $H(f)$ and $R(f)$ highlights the hidden structure of the problem which a pattern recognition system has to be able to detect, at least partially.

References

[1] Abu-Mostafa, Y. S. (1985), Complexity of random problems, *in* "Abstracts of Papers, IEEE International Symposium on Information Theory, Brighton, England," IEEE Catalog # 85 CH 2201-2, 84.

[2] Abu-Mostafa, Y. S. (1986), The complexity of information extraction, *IEEE Trans. on Information Theory* **IT-32**, 513-525.

[3] Abu-Mostafa, Y. S., Psaltis, D. (1987), Optical neural computers, *Scientific American* **256**, Number 3, 88-95.

[4] Chaitin, G. J. (1982), Gödel's theorem and information, *International Journal of Theoretical Physics* **22**, 941-954.

[5] Kolmogorov, A. N. (1965), Three approaches for defining the concept of information quantity, *Information Transmission* **1**, 3-11.

[6] Martin-Löf, P. (1966), The definition of random sequences, *Information and Control* **9**, 602-619.

[7] Pippenger, N. (1977), Information theory and the complexity of Boolean functions, *Mathematical Systems Theory* **10**, 129-167.

[8] Shannon, C. E. (1948), A mathematical theory of computation, *Bell System Technical Journal* **27**, 379-423.

[9] Savage J. E. (1976), "The Complexity of Computing," Wiley-Interscience, 14-66.

[10] Traub, J. F. (1985), Complexity of approximately solved problems, *Journal of Complexity* **1**, 3-10.

[11] Turing, A. M. (1936), On computable numbers with an application to the Entscheidungsproblem, *Proc. London Mathematical Society* **42**, 230-265.

2 Higher Order Associative Memories and their Optical Implementations

The properties of higher order memories are described. The non-redundant, up to N-th order polynomial expansion of N-dimensional binary vectors is shown to yield orthogonal feature vectors. The properties of expansions that contain only a single order are investigated in detail and the use of the sum of outer product algorithm for training higher order memories is analyzed. Optical implementations of quadratic associative memories are described using volume holograms for the general case and planar holograms for shift invariant memories.

2.1 Introduction

An associative memory can be thought of as a system that stores a prescribed set of vector pairs $(\underline{x}^m, \underline{y}^m)$ for $m = 1, \ldots, M$ and also produces $\underline{y}^m$ as its output when $\underline{x}^m$ becomes its input. We denote by N and N_0 the dimensionality of the input and output vectors respectively. When the output vectors are stored as binary N_0-tuples, the associative memory can be implemented as an array of discriminant functions, each dichotomizing the input vectors into two classes. This type of associative memory is shown schematically in Fig. 1. In evaluating the effectiveness of a particular associative memory we are concerned with its ability to store a large number of associations (capacity), the ease with which the parameters of the memory can be set to realize the prescribed mappings (learning), and how it responds to inputs that are not members of its training set (generalization). In this paper we discuss a class of associative memories known as higher order memories that have been recently investigated by a number of separate research groups[1,2,3,4,5,6,7,8]. Our motivation for investigating these memories was the increase in storage capacity that results from the increase in the number of independent papameters or degrees of freedom that is needed to describe a higher order associative mapping. The relationship between the degrees of freedom of a memory and its ability to store associations[9] is fundamental to this work and we state it in the following subsection as a theorem.

2.1.1 Degrees of freedom and storage capacity

Let D be the number of independent variables (degrees of freedom) we have under our control to specify input-output mappings and let each parameter have K separate levels or values that it can assume. We define the storage capacity C to be the maximum number of arbitrary associations that can be stored and recalled without error.

Theorem 1

$$C \leq \frac{D \log_2 K}{N_0}. \tag{1}$$

Proof : The number of different states of the memory is given by K^D and the total number of outputs that a given set of M input patterns can be mapped to is $2^{N_0 M}$. If the number of mappings were larger than the number of distinct memories then mappings would exist that are not implementable. Requiring that all mappings can be done leads to the relationship of the theorem.

The equality in Eq. 1 is achieved by Boolean circuits such as programmable logic arrays and an extreme case of a higher order memory we will discuss later. When the equality holds, resetting any one bit in any one of the parameters of the memory gives a different mapping. Such a memory cannot learn from the training set to respond in some desirable way to inputs that it has never seen before. The only way to get generalization when $C = D \log_2 K/N_0$ is to impose on it the overall structure of the memory before learning begins. One of the appealing features of neural architectures is the considerable redundancy in the degrees of freedom that is typically available. Therefore, there is hope that while a memory learns specific input-output correspondances it can also discover the underlying structure that may exist in the problem and learn to respond correctly for a set of inputs much larger than the training set. Moreover the same redundancy is responsible for the error tolerance that is evident in many neural architectures. Higher order memories are generally redundant and they can provide us with a methodology for selecting the degree of redundancy along with the number of degrees of freedom and the associated capacity to store random problems.

It is important to keep in mind that Eq. 1 holds for *arbitrary* mappings. If the input and output vectors are restricted in some way that happens to be matched to the architecture of a particular associative memory then it may be possible to overcome this limit. However, selecting the architecture of the associative memory such that it optimally implements only a subset of all possible associations is basically equivalent to choosing the architecture so that it generalizes in a desirable way. For instance suppose that we design an associative memory so that it is shift invariant (i.e. the output is insensitive to a change in the position of the input)[4,10]. Then this system will respond predictably to all the shifted versions of the patterns that were used to train it. We can equivalently think of this system as having a larger storage capacity than the limit of Eq. 1 over the set of shift invariant mappings. If we can identify *a priori* the types of generalization we wish the memory to exhibit and we can find ways to impose these on the architecture then this is certainly a sensible thing to do. Higher order memories can also provide a convenient framework within which this can be accomplished.

The penalty we must pay for the increase in the storage capacity that is afforded by the increase in the degrees of freedom in a higher order assocative memory is increase in implementation complexity. The computer that implements a higher order memory must have sufficient storage capacity to store a very large number of parameters. Moreover it must be capable of addressing the stored information with a high degree of parallelism in order to produce an output quickly. We will discuss in this paper optical implementations of second order memories and we will show a remarkable compatability between the computational requirements of these memories and the ability of optics to store information in three dimensions.

2.1.2 Linear discriminant functions and associative memories

We will consider as a precursor the most familiar associative memories that are constructed as arrays of linear discriminant functions[14]. A linear discriminant function is a mapping from the sample space X, a subset of R^N, to 1 or -1.

$$y = sgn\{\underline{w}^t \cdot \underline{x} + w_0\}$$

$$= sgn\{w_0 + w_1x_1 + w_2x_2 + \cdots + w_Nx_N\} \quad (2)$$

where *sgn* is the signum function, $\underline{w}$ is a weighting vector and w_0 is a threshold value. In this case the capacity is upperbounded by $(N+1)\log_2 K$ according to our definition of capacity. In this relatively simple case the exact capacity is known to be equal to $C = N + 1$ assuming the input points are in general position and $K = \infty$[11]. An associative memory is constructed by simply forming an array of linear discriminant functions each mapping the same input to a different binary variable. Several algorithms exist for training such memories including the perceptron, Widrow-Hoff, sum of outer products, pseudoinverse, and simplex methods[13,14,15,16] This memory can be thought of as the first order of the broader class of higher order memories that contain not only a linear expansion of the input vector but also quadratic and higher order terms. We will see in section 3 that the learning methods that are applicable to the linear memories generalize direcly to the higher order memories. First however we will describe the properties of the mappings that are implementable with higher order memories in section 2. Finally, in section 4 we will describe optical implementations of quadratic optical memories[2,12].

2.2 Properties of higher order memories

A Φ-function is defined to be a *fixed* mapping of the input vector $\underline{x}$ to an L-dimensional vector $\underline{z}$ followed by a linear discriminant function.

$$\begin{aligned} y &= sgn\{\underline{w}'^{t} \cdot \underline{z}(\underline{x}) + w_0\} \\ &= sgn\{w'_1z_1 + w'_2z_2 + \cdots + w'_Lz_L + w_0\} \end{aligned} \quad (3)$$

where $\underline{z}(\underline{x}) = (z_1(\underline{x}), z_2(\underline{x}), \ldots, z_L(\underline{x}))$, $\underline{w}'$ is an L dimensional weighting vector and $\underline{z}(\underline{x})$ is an L dimensional vector derived from $\underline{x}$. The storage capacity in this case is equal to the capacity of the second layer $L+1$[11] if the samples $\underline{z}$ are in general position whereas the upper bound on the capacity from Eq. 1 is $(L+1)\log_2 K$. The inefficiency in this case is $\log_2 K$ bits, the same as for the linear discriminant function even though the capacity can be raised arbitrarily by increasing L. It is not known what the exact relationship between L and K is. I.e. we do not know whether for higher

dimensions we need better resolution for the values of the weights to be capable of implementing a fixed fraction of the linear mappings. Recently Mok and Psaltis[17] have found the asymptotic (large N) statistical capacity to be $C = N$ for a linear discriminant function with binary weights. This result implies that even for large N for the vast majority of linear dichotomies, a large number of levels is not required. Therefore a Φ-function is an effective and straightforward method for increasing the capacity of an associative memory without loss in efficiency.

A higher order associative memory is an array of Φ-functions with the mappings $\underline{z}(\underline{x})$ being polynomial expansions of the vector $\underline{x}$. The schematic diagram of a higher order associative memory is shown in Fig. 2. When the polynomial expansion is of the r-th order in $\underline{x}$ then the output vector $\underline{y}$ is given by

$$y_l = sgn\{W_l^r(\underline{x}, \underline{x}, \ldots, \underline{x}) + W_l^{r-1}(\underline{x}, \ldots, \underline{x}) + \cdots, + W_l^2(\underline{x}, \underline{x}) + W_l^1\underline{x} + w_{l0}\} \tag{4}$$

where $l = 1, \ldots, N_0$, W_l^k is a k-linear symmetric mapping and W_l^1 is equivalent to $\underline{w}^t$ in Eq. 2. According to Eq. 3

$$z_j(\underline{x}) = x_{p_{j1}}^{n_1} x_{p_{j2}}^{n_2} \cdots x_{p_{jr}}^{n_r} \tag{5}$$

where $j = 1, 2, \ldots, L$, $p_{ji} \in \{1, 2, \ldots, N\}$ for $i = 1, \ldots, r$ such that all z_j are different and $n_1, n_2, \ldots, n_r = 0, 1$. Then L is $\binom{N+r}{r}$[11], and hence the capacity bound is $\left(\binom{N+r}{r} + 1\right)\log_2 K$ as before. For example, if $r = 2$, the function becomes quadratic and has the form $y_l = \underline{x}^t W_l^2 \underline{x} + W_l^1 \underline{x} + w_{l0}$ and the number of nonredundant terms in the quadratic expansion are $L = (N+1)(N+2)/2$.

The components of the vector $\underline{z}$ are binary-valued if $\underline{x}$ is binary. In this case, the samples cannot be assumed to be in general position since there are at most $N + 2$ binary vectors in N dimensional space which lie in general position. We will evaluate the effectiveness of higher order mappings in producing representations $\underline{z}(\underline{x})$ that are separable by the second layer of weights by calculating the hamming distance between $\underline{z}$ vectors given the hamming distance between the corresponding $\underline{x}$ vectors. We expect that if the Hamming distance between two binary vectors is large then they are easy to distinguish from one another.

2.2.1 Complete polynomial expansion of binary vectors

There are at most 2^N nonredundant terms in any polynomial expansion (Eq. 4) of a binary vector $\underline{x}$ in N dimensions. First, we will consider the following N-th order expansion (or equivalently bit production) for the bipolar vectors $\underline{x}$ in N dimensional binary space $\{1,-1\}^N$:

$$\underline{z} = \underline{z}(\underline{x}) = (1, x_1, x_2, \ldots, x_N, x_1x_2, \ldots, x_1x_2\ldots x_N)^t. \tag{6}$$

If we apply a linear discriminant function to the new vectors $\underline{z}$, then the capacity becomes 2^N which is equal to the total number of possible input vectors[2]. In other words this memory is capable of performing *any* mapping of N binary variables to any binary output vector $\underline{y}$. Of course the number of weights that are needed to implement this memory grows to 2^N times N_0, the number of bits at the output. In what follows we show that in this extreme case the vectors $\underline{z}$ become orthogonal to each other.

Theorem 2 *If we expand binary vectors $\underline{x}^m$ $(m = 1, 2, \ldots, 2^N)$ in $X_B = \{1,-1\}^N$ to 2^N dimensional binary vectors $\underline{z}^m$ according to Eq 6, where N is the dimensionality of the original feature vectors, then*
(a) $< \underline{z}^{m_1}, \underline{z}^{m_2} >= 2^N \delta_{m_1 m_2}$ *where* $< \cdot, \cdot >$ *is an inner product,*
(b) $\sum_i z_i^m = 0$,
(c) $\sum_m z_{j_1}^m z_{j_2}^m = 2^N \delta_{j_1 j_2}$ *and* $\sum_m z_j^m = 0$.

Example : The following table is for the case of $N = 3$. Note the orthogonality and the numbers of 1's and -1's in the new vectors and the set of each component of them except the first vector and the set of the first components.

x_1	x_2	x_3	1	x_1	x_2	x_3	x_1x_2	x_2x_3	x_3x_1	$x_1x_2x_3$
1	1	1	1	1	1	1	1	1	1	1
1	1	-1	1	1	1	-1	1	-1	-1	-1
1	-1	1	1	1	-1	1	-1	-1	1	-1
1	-1	-1	1	1	-1	-1	-1	1	-1	1
-1	1	1	1	-1	1	1	-1	1	-1	-1
-1	1	-1	1	-1	1	-1	-1	-1	1	1
-1	-1	1	1	-1	-1	1	1	-1	-1	1
-1	-1	-1	1	-1	-1	-1	1	1	1	-1

Table 1.

Proof : (a) Let us consider any two different binary vectors in the binary space of $\{1,-1\}^N$ whose Hamming distance is n $(1 \leq n \leq N)$. When they are expanded to two 2^N dimensional binary vectors, the number of k-th order terms that have opposite sign in the two expansions is

$$\binom{n}{1}\binom{N-n}{k-1} + \binom{n}{3}\binom{N-n}{k-3} + \binom{n}{5}\binom{N-n}{k-5} + \cdots\cdots\cdots \qquad (7)$$

Notice that two polynomials have different values if, and only if, they have odd number of terms whose signs are opposite. The Hamming distance between the two fully expanded (up to order 2^N) vectors can be calculated by adding the number of terms that have different sign over all the orders of the expansion:

$$\begin{aligned}
& \binom{n}{1}\binom{N-n}{0} + \binom{n}{1}\binom{N-n}{1} + \left\{\binom{n}{1}\binom{N-n}{2} + \binom{n}{3}\binom{N-n}{0}\right\} \\
+ & \left\{\binom{n}{1}\binom{N-n}{3} + \binom{n}{3}\binom{N-n}{1}\right\} + \cdots\cdots\cdots\cdots \\
+ & \left\{\binom{n}{1}\binom{N-n}{N-n} + \binom{n}{3}\binom{N-n}{N-n-2} + \binom{n}{5}\binom{N-n}{N-n-4} + \cdots\right\} \\
+ & \left\{\binom{n}{3}\binom{N-n}{N-n-1} + \binom{n}{5}\binom{N-n}{N-n-3} + \binom{n}{7}\binom{N-n}{N-n-5} + \cdots\right\} \\
+ & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
= & \sum_{i=odd}\binom{n}{i}\sum_{j=0}^{N-n}\binom{N-n}{j} = 2^{n-1}2^{N-n} \\
= & \ 2^{N-1}. \qquad (8)
\end{aligned}$$

The fact that the Hamming distance is 2^{N-1} for any two expanded vectors (for any n) proves that all of the 2^N vectors become orthogonal and that $< \underline{z}^{m_1}, \underline{z}^{m_2} >= 2^N \delta_{m_1 m_2}$.

(b) Just think of the cases where one of the two vectors is $(1,1,\ldots,1)$. Then, all the other vectors $\underline{z}$ have equal number of 1's and -1's because their Hamming distances are all 2^{N-1} from the $(1,1,\ldots,1)$ vector.

(c) See reference [13], page 109.

Slepian has discussed this orthogonalization property as a method for designing orthogonal codes and has given a different proof for it[18]. The proof presented here is useful for characterizing higher order memories because it allows us to trace the contribution of each order of the expansion to the orthogonalization and immediately derive results about the properties of quadratic and cubic memories. The output vector $\underline{y}$ is

$$y_l = sgn\{W_l \cdot \underline{z}\} = sgn\{\sum_{i=1}^{2^N} W_{li} z_i\} \tag{9}$$

where $l = 1, \ldots, N_0$ and W_l is a 2^N dimensional weighting row vector. The matrix W_{li} that can implement the $\underline{x}^m \mapsto \underline{y}^m$ mapping for $m = 1$ to 2^N can be formed in this case simply as the sum of outer products of $\underline{y}^m$ and $\underline{z}^m$:

$$W_{li} = \sum_{m=1}^{2^N} y_l^m z_i^m. \tag{10}$$

2.2.2 Expansions of a single order

The orthogonalization property of the full expansion is interesting because it shows that higher order memories provide a complete framework that takes us from the simplest "neuron", the linear discriminant function, to the full capability of a Boolean look-up table. Higher order memories can indeed provide a valuable tool for designing digital programmable logic arrays. In this paper, however, we are intersted in associative memories that are capable of accepting inputs with large N (e.g. if $N = 10^3$ then $2^N \approx 10^{300}$) in which case considering a full expansion of the input data is completely out of the question. In such cases we are really interested in an expansion that contains a large enough number of terms to provide the capacity needed to learn the problem at hand. In this subsection we analyze the properties of partial expansions that include all the terms of one order.

We will first consider the memory consisting of all the terms of a quadratic expansion with binary input vectors.

$$y_l = sgn\{\sum_i \sum_j w_{lij} x_i x_j\}$$

$$= sgn\{\sum_{k=1}^{L} w'_{ik} z_k\}. \tag{11}$$

The number of nonredundant terms in a quadratic expansion of a binary vector is $L = N(N-1)/2$. Let two input vectors have a Hamming distance n. The angle between these two vectors is given by the relation $\cos\theta_1 = 1 - (2n/N)$. The angle θ_2 between the corresponding $\underline{z}(\underline{x})$ vectors can be readily calculated since we know their Hamming distance from the proof of theorem 2(a):

$$\begin{aligned} \cos\theta_2 &= 1 - \frac{4n(N-n)}{N(N-1)} \\ &\approx 1 - 4\rho + 4\rho^2 = (1-2\rho)^2 \end{aligned} \tag{12}$$

where $\rho = n/N$. θ_2 and θ_1 are plotted versus ρ in Figure 3a. For $\rho < .5$, θ_2 is always larger than θ_1. Specifically for $\rho << 1$, $\theta_2 = \sqrt{2} \times \theta_1$. We see therefore that the quadratic mapping not only expands the dimensionality which provides capacity but also spreads the input samples apart, a generally desirable property. For $\rho > .5$ the quadratically expanded vectors are closer to each other than the original vectors and in the extreme case $n = N$, θ_2 becomes zero. This insensitivity of the quadratic mapping to a change in sign of all the bits is a property that is shared by all even order expansions. Next we consider a cubic memory

$$\begin{aligned} y_l &= sgn\{\sum_i \sum_j \sum_k w_{lijk} x_i x_j x_k\} \\ &= sgn\{\sum_{n=1}^{L} w'_{ln} z_n\} \end{aligned} \tag{13}$$

where $L = \binom{N}{3} + N$. In Figure 3b we plot θ_3, the angle between two cubically expanded binary vectors as a function of ρ. For convenience, θ_1 is also plotted in the same figure. In this case θ_3 increases faster with ρ for $\rho < .5$. For $\rho << 1$, $\theta_3 = \sqrt{3} \times \theta_1$. At $\rho \approx .4$ the cubic expansion gives essentially perfectly orthogonal vectors while for $\rho > .5$, θ_3 remains smaller than θ_1 and in the limit $\rho = 1$, $\theta_3 = \pi$. Thus the cubic memory discriminates between a vector and its complement.

The basic trends that are evident in the quadratic and cubic memories generalize to any order r. The number of independent terms in the r-th order expansion of a binary vector is $\binom{N}{r}$ which is maximum for $r \approx N/2$. Again this is not of practical importance because the number of terms in a full expansion of this sort is prohibitively large. What is of interest however is the effectiveness with which relatively small order expansions can orthogonalize a set of input vectors. The angle θ_r between two vectors that have been expanded to the r-th order is given by the following relation:

$$\cos\theta_r = \frac{\binom{N}{r} - 2\sum_{i=odd}\binom{n}{i}\binom{N-n}{r-i}}{\binom{N}{r}}. \tag{14}$$

We can obtain a simpler expression for the interesting case $r << N$ and for small ρ, $\theta_r \approx \sqrt{r} \times \theta_1$.

Proposition 3 *For $r << N$,*

$$\cos\theta_r \approx (1-2\rho)^r. \tag{15}$$

Moreover, for small ρ,

$$\theta_r \approx \sqrt{r}\theta_1 \tag{16}$$

where $\theta_1 \approx 2\sqrt{\rho}$.

Proof : For a small r, we can make the approximations $\binom{N}{r} \approx N^r/r!$, $\binom{n}{i} \approx n^i/i!$, and $\binom{N-n}{r-i} \approx (N-n)^{r-i}/(r-i)!$. Then, $\cos\theta_r$ is approximated as follows:

$$\begin{aligned}\cos\theta_r &\approx 1 - 2\sum_{i=odd}\frac{r!}{i!(r-i)!}\rho^i(1-\rho)^{r-i} \\ &= (1-2\rho)^r\end{aligned}$$

because of these relationships:

$$\sum_{i=odd} + \sum_{i=even} = (1-\rho+\rho)^r = 1,$$

$$\sum_{i=odd} - \sum_{i=even} = -(1-\rho-\rho)^r = -(1-2\rho)^r.$$

When $\rho << 1$, $\cos\theta_r$, which is approximately $1 - \theta_r^2/2!$, is approximated by $1 - 2r\rho$ directly from Eq. 14 or from Eq. 15. Therefore, it is followed by Eq. 16 that $\theta_r \approx 2\sqrt{r\rho}$.

We plot θ_r versus ρ for selected orders in Figure 4 using Eq. 15. It is evident that increasing r results in better separated feature vectors. Polynomial mappings act as an effective mechanism for increasing the dimensionality of the space in which inputs are classified because they guarantee a very even distribution of the samples in this new space.

2.3 Training of higher order memories

Once the initial polynomial mapping has been selected, the rest of the system in a higher order memory is simply a linear discriminant function. As such it can be trained by any of the existing methods for training linear discriminant functions. For instance the pseudoinverse[1,14,16] can be used to calculate the set of weights that will map a set of L-dimensional expanded vectors $\underline{z}^m$ to the associated output vectors $\underline{y}^m$. Alternatively, error driven algorithms such as the perceptron or adaline can be used to iteratively train the memory by repeatedly presenting the input vectors to the system, monitoring the output to obtain an error signal, and modifying the weights so as to gradually decrease the error. The relative ease with which higher order memories can be trained is a very important advantageous feature of this approach. A higher order memory is basically a multilayered network where the first layer is selected *a priori*. In terms of capacity alone, there is no advantage whatsoever in having multiple layers with modifiable weights. From theorem 1 we know that at best the capacity is determined by the number of modifiable weights. For a higher order memory we get the full advantage of the available degrees of freedom whereas if we put the same number of weights in multiple layers the resulting degeneracies will decrease the capacity. The relative advantage of trainable multiple layers is the potential for generalization that emerges through the learning process. The generalization properties of higher order memories on the other hand are mostly determined by the choise of the terms used in the polynomial expansion in the fixed first layer. Thus the generalization properties of these memories as described in this paper are imposed *a priori* by the designer

of the system.

The sum of outer products algorithm that has been used extensively for training linear associative memories can also be used for training the higher order memories and this algorithm generalizes to the higher order case in particularly interesting ways. In addition this particular learning algorithm is predominantly used for the holographic optical implementations that are described in the following section. Therefore we will discuss in some detail the properties of higher order memories that are trained using this rule.

2.3.1 The outer product rule

Let us consider associative memories constructed as an expansion of the r-order only with input samples in an N dimensional binary space and $r \geq 1$. Then

$$y_l = sgn\{\sum_{j_1 j_2 \cdots j_r} W_{l j_1 j_2 \cdots j_r} x_l x_{j_1} x_{j_2} \cdots x_{j_r}\} \tag{17}$$

where $1 \leq j_1, j_2, \cdots, j_r \leq N$, $1 \leq l \leq N_0$. The number of independent terms L in the r-th order expansion is $\binom{N+r-1}{r}$ which for $r << N$ can be approximated by $N^r/r!$

The expression for the weights of the r-th order expansion using the sum of outer products algorithm[2,3] is

$$W_{l j_1 j_2 \cdots j_r} = \sum_{m=1}^{M} y_l^m x_{j_1}^m x_{j_2}^m \cdots x_{j_r}^m \tag{18}$$

where M is the number of vectors stored in the memory, $\underline{y}^m$ is an output vector associated with an input vector $\underline{x}^m$ as before. With the above expression for the weight tensor Eq. 17 can be rewritten as follows

$$y_l = sgn\{\sum_{m=1}^{M} y_l^m (\sum_{j=1}^{N} x_j^m x_j)^r + w_l^0\}. \tag{19}$$

The above equation suggests an alternate implementation for higher order memories that are trained using the outer product rule. This is shown schematically in Fig. 5. The inner products between the input vector and

all the stored vectors $\underline{x}^m$ are formed first, then raised to the r-th power, and the signal from the m-th unit is connected to the output through interconnective weights y_i^m. If $\underline{y}^m = \underline{x}^m$ then the memory is autoassociative, and in this case the output can be fed back to the input resulting in a system whose stable states are programmed to be the vectors $\underline{x}^m$. This becomes a direct extension of the Hopfield network[15,19,20] to the higher order case. Assuming that $\underline{x} = \underline{x}^n$ is one of the stored vectors, y_i becomes

$$\begin{aligned} y_i &= sgn\{N^r y_i^n + \sum_{m \neq n} y_i^m (\sum_{j=1}^{N} x_j^m x_j^n)^r + w_i^0\} \\ &= sgn\{N^r y_i^n + n_i(\underline{x}^n)\} \end{aligned} \tag{20}$$

where the first term is the desired signal term, n_i is a noise term, and the thresholding weight is set to zero.

The expectation value of $n_i(\underline{x}^n)$ is zero if the bits that comprise the stored binary input and output vectors are drawn randomly and independently having equal probability of being $+1$ or -1. If this is the case then

$$E(\sum_{tt'} x_t^m x_{t'}^m) = \sum_{tt'} \delta_{tt'}, \quad E(\sum_{mm'} x_t^m x_t^{m'}) = \sum_{mm'} \delta_{mm'} \tag{21}$$

where $\delta_{i,j}$ is the Kronecker delta function. The variance of n_i is calculated as follows:

$$\begin{aligned} E(n_i^2) &= E(\sum_{m \neq n} \sum_{m' \neq n} y_i^m y_i^{m'} \sum_{j_1 j_2 \cdots j_r} \sum_{s_1 s_2 \cdots s_r} x_{j_1}^m x_{j_2}^m \cdots x_{j_r}^m x_{j_1}^n x_{j_2}^n \cdots x_{j_r}^n \\ &\qquad x_{s_1}^{m'} x_{s_2}^{m'} \cdots x_{s_r}^{m'} x_{s_1}^n x_{s_2}^n \cdots x_{s_r}^n) \\ &= E(\sum_{m \neq n} \sum_{j_1 j_2 \cdots j_r} \sum_{s_1 s_2 \cdots s_r} x_{j_1}^m x_{j_2}^m \cdots x_{j_r}^m x_{s_1}^m x_{s_2}^m \cdots x_{s_r}^m \\ &\qquad x_{j_1}^n x_{j_2}^n \cdots x_{j_r}^n x_{s_1}^n x_{s_2}^n \cdots x_{s_r}^n). \end{aligned} \tag{22}$$

In the above we used the facts that different stored vectors are uncorrelated (i.e. for $m \neq m'$) and $y_i^2 = 1$. Then, the variance becomes $(M-1)Q(N,r)$, where $Q(N,r)$ is the number of possible permutations such that

$$\delta_{i_1 t_1} \delta_{i_2 t_2} \cdots \delta_{i_r t_r} = 1 \tag{23}$$

where the set of variables $\{i_1, \ldots, i_r, t_1, \ldots, t_r\}$ spans all the combinations produced by the set of variables $\{j_1, \ldots, j_r, s_1, \ldots, s_r\}$. The variance can

be calculated exactly for the cases r=1,2 and 3 and it is $(M-1)N$, $(M-1)(3N^2-2N)$ and $(M-1)(15N^3-30N^2+16N)$, respectively. For the general case we will derive lower and upper bounds which for large N provide us with a good estimate of the variance for any order r.

Proposition 4 *The total number of permutations, $Q(N,r)$, for which Eq. 23 holds, satisfies the following relationship:*

$$P(N,r)\frac{(2r)!}{2^r r!} + \binom{2r}{4} P(n,r-1)\frac{(2r-4)!}{2^{r-2}(r-2)!} \leq Q(N,r) \leq N^r \frac{(2r)!}{2^r r!} \quad (24)$$

where $P(m,n) \equiv m!/(m-n)!$.

Proof : The number of ways of making r pairs of $2r$ items is $(2r-1)(2r-3)\cdots(3)(1) = (2r)!/2^r r!$. The items that we are concerned with are the variables i_j, t_j and each of these variables can take one of N values. We can only select the values of half these variables (N^r possibilities) and for each of these choises we can create r pairs. Hence the upperbound is $N^r(2r)!/2^r r!$. This is an upper bound because we have overcounted for different pairings of variables that have the same value.

The initial lower bound is derived if each pair has a different value from all others, which eliminates the possibility of overcounting. The number of possible ways to satisfy Eq. 23 with the variables in any two pairs not taking the same values is $P(N,r)(2r)!/2^r r!$. This is an underestimate because all pairs that contain variables taking the same value should be counted once. We can thus improve the lower bound by counting the number of ways these degenerate pairings occur and adding them into the previous bound. For example when two pairs out of r have the same values with $\binom{2r}{4}$ choices, there are $\binom{2r}{4} NP(N-1,r-2)(2r-4)!/2^{r-2}(r-2)!$ possible permutations where $(2r-4)!/2^{r-2}(r-2)!$ is the number of ways of making $r-2$ pairs of $2r-4$ items. Therefore, $Q(N,r)$ is lower bounded by $P(N,r)(2r)!/2^r r! + \binom{2r}{4} P(N,r-1)(2r-4)!/2^{r-2}(r-2)!$, since $NP(N-1,r-2) = P(N,r-1)$.

We can get a very good approximation to the SNR using the approximations of $M-1 \approx M$ and $Q(N,r) \approx N^r(2r)!/2^r r!$ which are very nearly true for the interesting case $r << N$:

$$SNR \approx \frac{N^r}{\{MN^r(2r)!/2^r r!\}^{1/2}}$$

$$= \left\{\frac{N^r}{M}\frac{2^r r!}{(2r)!}\right\}^{1/2}. \tag{25}$$

For example, the linear memory, $r = 1$, has a $SNR \approx (N/M)^{1/2}$, the quadratic memory, $r = 2$, a SNR of $N/(3M)^{1/2}$ and the cubic memory, $r = 3$, a SNR of $(N^3/15M)^{1/2}$. We can obtain an estimate for the capacity of an r-th order memory by equating the signal to noise ratios of the linear and r-th order memories and solving for M_r, the number of stored vectors that will yield the equality. For r small compared to N we obtain

$$\frac{M_r}{M_1} = N^{r-1}\frac{2^r r!}{(2r)!} \tag{26}$$

Comparing its value with the capacity M_1 of a linear memory we can obtain the relationship between the capacities, that is, $M_r/M_1 = N^{r-1}2^r r!/(2r)!$. For example M_2 of a quadratic memory is $M_1 N/3$ and M_3 of a cubic memory is $M_1 N^2/15$.

The diagonal terms in a high order memory $W_{l j_1 j_2 \cdots j_r}$ can be defined as those of which all the indexes j are not different. We form the weight tensor with zero diagonal as follows:

$$W_{l j_1 j_2 \cdots j_r} = \begin{cases} \sum_m y_l^m x_{j_1}^m x_{j_2}^m \cdots x_{j_r}^m & \text{if } j\text{'s are all different,} \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

When the input is one of the stored vectors $\underline{x}^n$ and the weight tensor has zero diagonal, the output y_l becomes

$$\begin{aligned} y_l &= sgn\{\sum_{different\ j} W_{l j_1 j_2 \cdots j_r} x_{j_1}^n x_{j_2}^n \cdots x_{j_r}^n + w_l^0\} \\ &= sgn\{P(N,r) y_l^n + \sum_{m \neq n} y_l^m \sum_{different\ j} x_{j_1}^m x_{j_2}^m \cdots x_{j_r}^m x_{j_1}^n x_{j_2}^n \cdots x_{j_r}^n + w_l^0\} \end{aligned} \tag{28}$$

where the first term is a signal term and the second a noise term as before. The variance of the noise term is easily shown to be $(M-1)P(N,r)r!$ using Eq. 21. Therefore, the SNR becomes

$$SNR = \left\{\frac{P(N,r)}{(M-1)r!}\right\}^{1/2} \approx \left\{\binom{N}{r}/M\right\}^{1/2} \tag{29}$$

which can be approximated as $(N^r/Mr!)^{1/2}$ for $r << N$.

Chen and his coworkers[3] introduced an energy function[15,21] for the r-th order autoassociative memory with feedback and outer products as follows:

$$E_r = -\sum_{m=1}^{M} < \underline{x}^m, \underline{x} >^{r+1} \tag{30}$$

where $< \cdot, \cdot >$ denotes an inner product of two vectors. The change in the energy due to a change $\delta\underline{x}$ in the state of the network was shown by Chen et. al. to be decreasing for odd r.

$$\begin{aligned} \Delta E_r &\equiv E_r(\underline{x} + \delta\underline{x}) - E_r(\underline{x}) \\ &= -(r+1)\sum_l \delta x_l \sum_{j_1 \cdots j_r} W_{l j_1 j_2 \cdots j_r} x_{j_1} x_{j_2} \cdots x_{j_r} - R_r \end{aligned} \tag{31}$$

where

$$R_r \equiv \sum_m \sum_{j=2}^{r+1} \binom{r+1}{j} < \underline{x}^m, \underline{x} >^{r+1-j} < \underline{x}^m, \delta\underline{x} >^j . \tag{32}$$

The first term in Eq. 31 is always nonpositive because of the specification of the update rule: $\delta x_l \geq 0$ if $\sum_{j_1 \cdots j_r} W_{l j_1 j_2 \cdots j_r} x_{j_1} x_{j_2} \cdots x_{j_r} \geq 0$ and vice versa. They showed that the second term is also nonpositive by showing that R_r is an increasing function of r for r odd and $R_1 > 0$.

For r even it is possible to prove that the autoassociative memory converges only for asynchronous updating even though in simulations even order autoassociative memories consistently converge as well. The fact that the energy is not always decreasing when r is even may actually be helpful for getting out of local minima and settling in the programmed stable state which are global minima in a region of the energy surface. A descent procedure that is always decreasing in energy cannot escape local minima since there is no mechanism for climbing out of them. As an example, consider a quadratic memory, i.e. r=2 (even), whose energy function is given by

$$E_2 = -\sum_{ijk} W_{ijk} x_i x_j x_k \tag{33}$$

$$\Delta E_2 = -3\sum_{ijk} W_{ijk} x_j x_k \delta x_i - 3\sum_{ijk} W_{ijk} x_k \delta x_i \delta x_j - \sum_{ijk} W_{ijk} \delta x_i \delta x_j \delta x_k. \tag{34}$$

The first term is nonincreasing but the second and third terms can be *increasing*. If the vector $\underline{x}$ is very close to one of the stored vectors $\underline{x}^n$ then the first term becomes dominant and the energy will be very likely to be nonoincreasing causing the system to settle at $\underline{x} = \underline{x}^n$. If $\underline{x}$ is not close to any of the stored vectors, then all three terms in the above equations are on the average comparable to each other and since two of them are not nondecreasing the energy function may be increasing and it is possible to escape from local minima.

2.4 Optical implementations of quadratic associative memories

The outer product quadratic associative memories described in the previous section require three basic components for their implementation: interconnective weights, a square-law device, and a threshold nonlinearity. In this section, we present a variety of optical implementations using either planar or volume holograms to provide the interconnection pathways and optical or electro-optical devices to provide the required nonlinearities.

Since holographic techniques are used to implement the required interconnections, we will first briefly discuss holography[22] and in particular the distinction between the use of planar versus volume holograms. The holographic process is shown schematically in Fig. 6. In the recording step (Fig. 6a) the interference between the reference plane wave that is created by collimating the light from a point source using a lens and the wave originating from the object "A" is recorded on a planar light sensitive medium such as a photographic plate. When the developed plate is illuminated with the same reference wave, the field that is diffracted by the recorded interference pattern gives a virtual image of the original object which can be converted to a real image with a lens. The reconstruction of the hologram is thus equivalent to interconnecting the single point from which the plane wave reference is derived to all the points that comprise the reconstucted image. The weight of each interconnection is specified by the interference pattern stored in the hologram.

Volume holograms are prepared and used in the same manner except that whereas a planar hologram records the interference pattern as a two

dimensional pattern on a plane, a volume hologram records the interference pattern throughout the volume of a three dimensional medium. The disparity in the dimensionalities of the two storage formats results in marked differences in the capabilities of the two processes. This difference is explained with the aid of Figs. 7a and 7b where the reconstruction of both a planar and a volume hologram are shown. Each hologram is prepared to store the two images "A" and "O" by double exposure with each image being associated with a reference plane wave that is incident on the hologram at a different angle. Each reference plane wave is generated by a separate point source and thus the reconstuction of a hologram with the two reference waves is equivalent to interconnecting multiple input points to all the points on the plane of the reconstucted image. In the case of the planar hologram, however, when either one of the reference waves is incident both images are reconstructed. This implies that we cannot in this case independently specify how each of the input points is connected to the output. In contrast, because of the interaction of the fields in the third dimension[23] the volume hologram is able to resolve the differences in the angle of incidence of the reference beam and upon reconstruction when the reference for "A" illuminates the medium, only "A" is reconstructed and similarly for the second pattern. When both input points are on simultaneously then each is interconnected to the output independently according to the way it was specified by the recording of the two holograms. Thus volume holograms provide more flexibility for implementing arbitrary interconnections which translates to efficient three dimensional storage of the interconnective weights needed to specify the quadratic memory.

Another way in which we can draw the distinction between planar and volume holograms is in terms of the degrees of freedom. The implementation of a quadratic memory whose input word size is N bits requires approximately N^3 interconnections for the three dimensional interconnection tensor. The number of degrees of freedom of the planar hologram of area A is upper bounded by A/δ^2 while that of a volume hologram is limited to V/δ^3, where V is the volume of the crystal and δ is the minimum detail that can be recorded in any one dimension[24,25,26]. Equating the degrees of freedom that are required to do the job to those that are available, the crystal volume is determined to be at least $V = N^3\delta^3$ whereas a planar

hologram to do the same job would require a hologram of area $A = N^3\delta^2$. For comparison, a network with $N = 10^3$ can in principle be implemented using a cubic crystal with the length of each side being $l_v = N\delta$=1 cm, but a square planar hologram is required to have the length of each side be at least $l_p = N^{3/2}\delta$=0.33 m at $\delta = 10\ \mu m$. Thus, the volume hologram offers a more compact means of implementing large memory systems.

2.4.1 Volume hologram systems

There are several schemes for fully utilizing the interconnective capability of volume holograms[25,26]. For the implementation of quadratic memories we use volume holograms to fully interconnect a 2-D pattern to a 1-D pattern ($N^2 \mapsto N$ mappings) and also the reverse ($N \mapsto N^2$). The geometry for recording the weights for both cases is shown in Fig. 8a and the reconstruction geometries are illustrated in Figs. 8b and 8c. The circles represent the resolvable spots at the various planes in the system. The waves emanating from each point at the input planes are transformed into plane waves by the Fourier transform lenses L_1 and L_2 and interfere within the crystal, creating volume gratings.

The weights are loaded into the volume hologram with multiple holographic exposures in the system of Fig. 8a. In the following subsections we will describe several specific procedures for doing so. For the $N \mapsto N^2$ mapping (Fig. 8b) in reading out the stored information, a single source in the input array recontructs one of the N 2-D image consisting of N^2 pixels that it is associated with. The rest of the images, which belong to the other input points, are not read out because of the angular discrimination of volume holograms. The counterpart to this scheme, shown in Fig. 8c, which implements an arbitrary $N^2 \mapsto N$ mapping. This setup is basically the same as that of Fig. 8b except that the roles of the input planes have been interchanged or equivalently the direction in which light propagates has been reversed.

2.4.1a $N^2 \mapsto N$ schemes

First, we consider a method by which the full three dimensional interconnection tensor is implemented directly with a volume hologram. Recall that if the weight tensor is trained using the sum of outer products then it is

given by

$$w_{ijk} = \sum_{m=1}^{M} y_i^m x_j^m x_k^m, \tag{35}$$

where x_i^m represents the m-th input memory vector and y_i^m represents the associated output vector. Such a memory is accessed by first creating an outer product of the input vector and multiplying it with w_{ijk} as follows:

$$y_i = sgn\{\sum_{j=1}^{N}\sum_{k=1}^{N} w_{ijk} x_j x_k\}. \tag{36}$$

The volume hologram is prepared using the set-up in Fig. 8a. First, the outer product matrix of the m-th memory input vector, $x_j^m x_k^m$, is formed on an electronically addressed spatial light modulator (SLM)[27]. Another one dimensional SLM whose transmittance represents the m-th output vector y_i^m is placed in the other input plane, and the two SLMs are illuminated by coherent light. The transmitted waves are then Fourier transformed by lenses L_1 and L_2 to interfere within the crystal volume to create index gratings. This procedure is repeated for all M associated input-output pairs so that a sum of M holograms is created in the crystal. For the quadratic outer product memory whose capacity is fully expended, this involves on the order of $N^2/\log N$ exposures.

We will now describe another method for recording the weight vector in the volume hologram that involves fewer exposures and can also be used not only for the outer product scheme but for recording any given weight tensor as well. The same basic recording architecture of Fig. 8a is used in this case also. In the first exposure, the top light source in the linear array is turned on while the SLM is programmed with the matrix w_{1jk}, where w_{ijk} is the interconnection tensor. When the SLM is illuminated with light coherent with that of the point source, the crystal records the mutual interference pattern as a hologram of the image w_{1jk} with a reference beam that is the plane wave generated from the top light source. In the next step, the second source is turned on while the SLM is programmed with the matrix w_{2jk}. In this manner the connectivity for all the points in the linear array at the input are sequentially specified and the memory training is completed when all N exposures have been made. The disadvantage of this method relative

to the outer product recording is the need to precalculate electronically the weight tensor but it has the advantage of fewer exposures (N versus $N^2/\log N$) and greater flexibility in choosing the training method.

The architecture in Fig. 8c is used to access the data stored in the hologram by either one of the recording methods described above. The electronically addressed 2-D SLM is placed at the input plane and it is programmed with the outer product matrix $x_k x_j$ of the input vector. The light from the N^2 input points is interconnected with the N output points via the recorded w_{ijk} interconnect kernel. A linear array of N photodetectors is positioned to sample the output points.

It is important to restate at this juncture that this particular implementation achieves the quadratic interconnections by first transforming the N input features (i.e., the N elements of the input vector x_j) into a set of N^2 features via the outer product operation. The result is that although the interconnections are quadratic with respect to the N original feature points, they are linear with respect to the N^2 transformed features. This allows the application of error driven learning algorithms for linear networks such as the *Adaline*[28] where the interconnections are developed by an iterative training process[29]. The operation of such a learning scheme is illustrated in Fig. 9 which is the same basic architecture as Fig. 8c with feedback from the output back into one of the input ports. Each iteration consists of a reading and a writing phase. During the reading phase, the interconnections present in the crystal are interrogated with a particular item to be memorized by illuminating the 2-D SLM which contains the outer product matrix $\underline{x}^m \underline{x}^{mt}$ and the output is formed on the detector array. In the subsequent writing phase, the error pattern generated by subtracting the actual output from the desired output pattern is loaded into the 1-D SLM and both SLM's (the 2-D SLM still contains $\underline{x}^m \underline{x}^{mt}$) are illuminated with coherent light, forming a set of gratings in addition to the previously recorded gratings. The procedure is iteratively repeated for each item to be memorized until the output error is sufficiently small. This algorithm is a descent procedure designed to minimize the mean squared cost $\epsilon = \frac{1}{M} \sum_{m=1}^{M} \left[\sum_{j=1}^{N} \sum_{k=1}^{N} w_{ijk} x_j^m x_k^m - y^m \right]^2$ by iteratively updating the interconnection values.

2.4.1b $N \mapsto N^2$ schemes

The $N \mapsto N^2$ mapping capability of the volume hologram which is the inverse of that required for the architectures just described can be used also to implement quadratic memories and can be generalized for higher order memories. The basic idea behind this scheme is illustrated in Fig. 10 which shows the interconnection between the i-th and j-th neurons whose weight w_{ij} is a linear combination of all of the inputs and is described by

$$w_{ij} = \sum_{k=1}^{N} \hat{w}_{ijk} x_k. \tag{37}$$

The overall result is, of course, recognized to be the equation describing the quadratic memory, but the notion of an input dependent weight suggests the implementation shown in Fig. 11. The system is basically an optical vector matrix multiplier[30] in which the matrix is created on an optically addressed SLM by multiplying the input vector with the three dimensional tensor stored in a volume hologram. The input vector is represented by a one dimensional array of light sources. The portion of the system on the left side of the SLM is the vector matrix multiplier and it works as follows. Light from each input point is imaged horizontally but spread out vertically so that each source illuminates a narrow, vertical area on the 2-D SLM. The reflectance of the SLM corresponds to the matrix of weights w_{ij} in Eq. 37. The reflected light from the SLM travels back towards the input and a portion of it is reflected by a beam splitter and then imaged horizontally but focused vertically onto a 1-D output detector array. The output from the detector array represents the matrix vector product between the input vector and the matrix represented by the 2-D reflectance of the SLM. The matrix of weights, in this case, is not fixed but rather computed from the input via a volume hologram by exposing the righthand side of the SLM as shown in the figure. The optical system to the right of the 2-D SLM in Fig. 11 is the same as the $N \mapsto N^2$ system of Fig. 8b. The volume hologram which has been prepared to perform the appropriate dimension increasing operation ($N \mapsto N^2$), transforms the light distribution given by its one dimensional array of sources into the input dependent matrix of weights given by Eq. 37. This system is functionally equivalent to the previous system except it does not require the use of a 2-D electronically addressed input SLM. The 1-D devices utilized in this architecture are

easier and faster to use in practice. Instead a 2-D optically addressed SLM is needed which in practice is simpler to use compared to electonically addressed devices (requires less electronics), typically has more pixels, and potentially much higher speed. A disadvantage of this method however is that it does not lend itself for the direct implementation of the simple outer product training method without the use of an electronically addressed 2-D SLM.

The $N \mapsto N^2$ mapping technique can be used in conjunction with its inverse, the $N^2 \mapsto N$ mapping, to implement the quadratic outer product memory using two volume holograms, a 1-D electronically addressed SLM, and an optically addressed 2-D SLM. Shown in Fig. 12 is a schematic diagram of such a system. The first hologram is prepared with the multiple exposure scheme discussed earlier (Fig. 8a) where for each exposure, a memory vector in the one dimensional input array and one point in the two dimensional $(\sqrt{M} \times \sqrt{M})$ input training array are turned on simultaneously. The second hologram is prepared by a similar procedure except that the associated output vectors are recorded in correspondence to each point in the two dimensional training plane. After the holograms are thus prepared, an input vector is loaded into the one dimensional input array and the correlations between it and the M memory vectors are displayed in the output plane[31,32,33]. An optically addressed SLM can be used to produce an amplitude distribution which is the square of the incident correlation amplitudes. The processed light then illuminates the second hologram which serves as an $M \mapsto N$ interconnection, each correlation peak in the SLM plane reading out its corresponding memory vector and forming a weighted sum of the stored memories on the one dimensional output detector array. This is a direct optical implementation of the system shown in block diagram form in Fig. 5 with the 2-D SLM performing the square law nonlinearity at the middle plane and the two volume holograms providing the interconnections to the input and output.

2.4.2 Planar hologram systems

While not having the extra dimension to directly implement the three dimensional interconnection tensor for general quadratic memories, planar

holograms can nevertheless implement the outer product quadratic memory in a way similar to the one used in the system just described. The planar holographic system is shown in Fig. 13. Here, the information is stored in the two multichannel 1-D Fourier transform (FT) holograms, the first of which contains the 1-D FT's of the M memory input vectors and the other, the FT's of the associated output vectors[10]. The first part of the system is a multichannel correlator which correlates the input against each of the M memory vectors. At the correlation plane, the M correlation functions stacked up vertically are sampled at $x = 0$ with a slit to obtain the required inner products which are then squared by the SLM. Each resulting point source of light is then collimated horizontally and imaged vertically onto the second hologram to illuminate that portion which contains the corresponding output vector. The final stage computes the FT of the light distribution just following the second hologram to produce the weighted sum of the vectors at the output detector array. It is interesting to note that if the SLM is removed from the correlation plane, this system reduces to the linear outer product memory.

Notice that in this system if the input pattern shifts horizontally then the correlation peak also shifts in the correlation plane and it is blocked by the slit that is placed there. Therefore shifted versions of the input vector are not recognized, as expected. Shift invariance where the shifted versions of the memory vectors are recognized and their associated outputs, shifted by the same amount as the input, are retrieved can be built into this sytem by simply lengthening the input SLM and the output detector array to accomodate the shifts and removing the slit in the correlation plane. The resulting system treats each of the $2N - 1$ shifted versions of the memory vectors as a new memory and as a result, the increased capacity of the quadratic memory over the linear one (by a factor of N) is expended to provide invariant operation.

References

[1] T. Poggio, "On Optimal Nonlinear Associative Recall," Biol. Cybernetics, 19, p. 201, 1975.

[2] D. Psaltis and C.H. Park, "Nonlinear Discriminant Functions and Associative Memories," J. Denker, ed., AIP Confer. Procs. 151, p. 370, Snowbird, UT 1986.

[3] H.H. Chen, Y.C. Lee, T. Maxwell, G.Z. Sun, H.Y. Lee, and C.L. Giles, "High Order Correlation Model for Associative Memory," J. Denker, ed., AIP Confer. Procs. 151, p. 86, Snowbird, UT 1986.

[4] T. Maxwell, C.L. Giles, Y.C. Lee, and H.H. Chen, "Nonlinear Dynamics of Artificial Neural Systems," J. Denker, ed., AIP Confer. Procs. 151, p. 299, Snowbird, UT 1986.

[5] T. Sejnowski, "High-Order Boltzmann Machines," J. Denker, ed., AIP Confer. Procs. 151, p. 398, Snowbird, UT 1986.

[6] P. Baldi, and S.S. Venkatesh, "Number of Stable Points for Spin-Glasses and Neural Networks of Higher Orders," Phys. Rev. Lett., 58(9), p. 913, 1987.

[7] C.M. Newman, "Memory Capacity in Symmetric Neural Networks: Rigorous Bounds," presented in Neural Network Conf. in Denver, Colorado, 1987.

[8] C.L. Giles and T. Maxwell, "Learning and Generalization in Higher Order Networks," Appl. Opt., Dec. 1987.

[9] Y. Abu-Mostafa and D. Psaltis, "Computation Power of Parallelism in Optical Architectures," IEEE Comp. Soc. Workshop on Comp. Arch. for Patt. Anal. and Im. Database Manag., Miami Beach, Fl, p. 42, Nov. 1985.

[10] D. Psaltis and J. Hong, "Shift-Invariant Optical Associative Memories," Opt. Engr., 26(1), p. 10, 1987.

[11] T.M. Cover, "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition," IEEE Trans. Elec. Comp., EC-14, p. 326, 1965.

[12] D. Psaltis, C.H. Park, and J. Hong, "Quadratic Optical Associative Memories,", JOSA A, 3(13), p. 32, 1986.

[13] R. Duda and P. Hart, "Pattern Classification and Scene Analysis," Wiley, N-Y, 1973.

[14] T. Kohonen, "Self Organization and Associative Memory," Springer-Verlag, N-Y, 1984.

[15] J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proc. Acad. Sci, USA, 79, p. 2554, 1982.

[16] S.S. Venkatesh and D. Psaltis, "Linear and Logarithmic Capacities of Associative Memories," to be published in IEEE Trans. Infor. Th..

[17] F. Mok and D. Psaltis, in preparation.

[18] D. Slepian, "A Class of Binary Signaling Alphabets," Bell Sys. Tech. Jour., p. 203, 1956.

[19] J.A. Anderson, "Cognitive and Psychological Computation with Neural Models," IEEE Trans. Sys., Man, and cyber., SMC-13, p. 799, 1983.

[20] K. Nakano, "Association-A Model of Associative Memory," IEEE Trans. Sys., Man and Cyber., SMC-2(3), 1972.

[21] M. Cohen and S. Grossberg, "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks," IEEE Trans. Sys., Man and Cyber., SMC-13, p. 815, 1983.

[22] R.J. Collier, C.B. Burkhardt, and L. H. Lin, Optical Holography, Academic Press, N-Y, 1971.

[23] H. Kogelnik, "Coupled Theory for Thick Hologram Gratings," BSTJ, 48, p. 2909, 1969.

[24] P.J. Van Heerden, "Theory of Optical Information Storage in Solids," Appl. Opt., 2(4), p. 393, 1963.

[25] D. Psaltis, J. Yu, X.G. Gu, and H. Lee, Second Topical Meeting on Optical Computing, Incline Village, NV, Mar. 1987.

[26] D. Psaltis and X.G. Gu, H. Lee, and J. Yu, "Optical Interconnections Implemented with Volume Holograms," to be published.

[27] C. Warde and A.D. Fisher, "Spatial Light Modulators: Applications and Functional Capabilities," in Optical Signal Processing, p. 477, J.L. Horner, ed., Academic Press, San Diego, 1987.

[28] B. Widrow and M.E. Hoff, "Adaptive Switching Circuits," IRE Wescon Conv. Rec., Pt. 4, p. 96, 1960.

[29] D. Psaltis, D. Brady, and K. Wagner, "Adaptive Optical Networks Using Photorefractive Crystals," to appear in Appl. Opt., Mar. 1988.

[30] J.W. Goodman, R.A. Dias, and L.M. Woody, "Fully Parallel, High Speed Incoherent Optical Method for Performing Discrete Fourier Transforms," Opt. Lett., 2(1), p. 1, 1978.

[31] E.G. Paek and D. Psaltis, "Optical Associative Memory Using Fourier Transform Holograms," Opt. Eng., 26(5), p. 428, May 1987.

[32] Y. Owechko, G.J. Dunning, E. Marom, and B.H. Soffer, "Holographic Associative Memory with Nonlinearities in the Correlation Domain," Appl. Opt., 26(10), p. 1900, 1987.

[33] R.A. Athale, H.H. Szu, and C.B. Friedlander, "Optical Implementation of Associative Memory with Controlled Nonlinearity in the Correlation Domain," Opt. Lett., 11(7), p. 482, 1986.

Figure Captions

Fig. 1 a. Discriminant function. b. Associative memory consructed as an array of discriminant functions.

Fig. 2 Higher order associative memory.

Fig. 3 a. The angle between linearly and quadratically expanded vectors as a function of the hamming distance at the input. b. The angle between linearly and cubically expanded vectors as a function of the hamming distance at the input.

Fig. 4 The angle between expanded vectors for selected orders.

Fig. 5 Outer product, r-th order associative memory.

Fig. 6 Holographic recording and reconstruction.

Fig. 7 Holographic interconnections using a.planar versus b. volume holograms.

Fig. 8 Optical interconnections using volume holograms. a. Recording apparatus. b. $N \mapsto N^2$ mapping. c. $N^2 \mapsto N$ mapping.

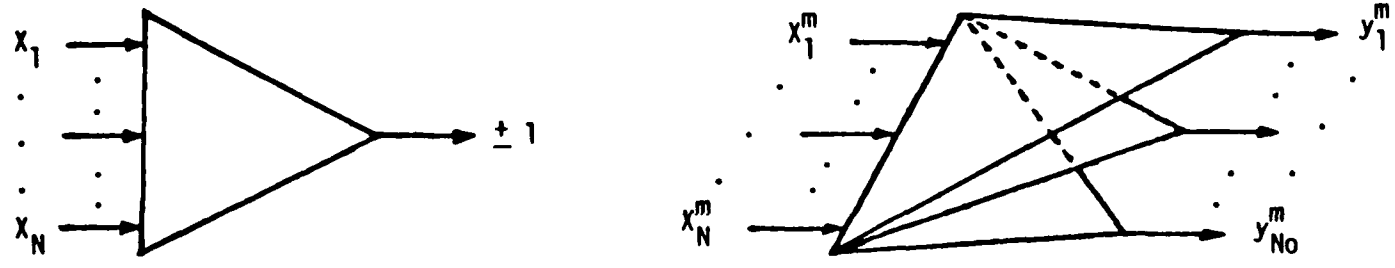
Fig. 9 Optical system for performing error driven learning in a higher order memory.

Fig. 10 Quadratic mappings implemented as nonnlinear interconnections.

Fig. 11 Optical architecture for the implementation of the nonlinear interconnections of Fig. 10.

Fig. 12 Optical higher order associative memory implemented with volume holograms.

Fig. 13 Optical implementation of the outer product higher order memory.



a. b.

Figure 1.

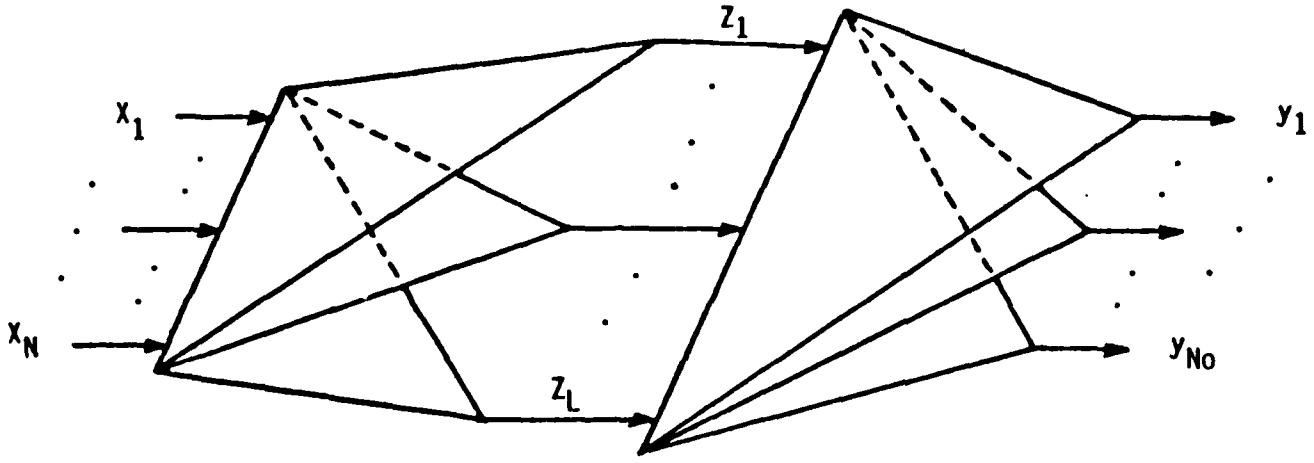


Figure 2.

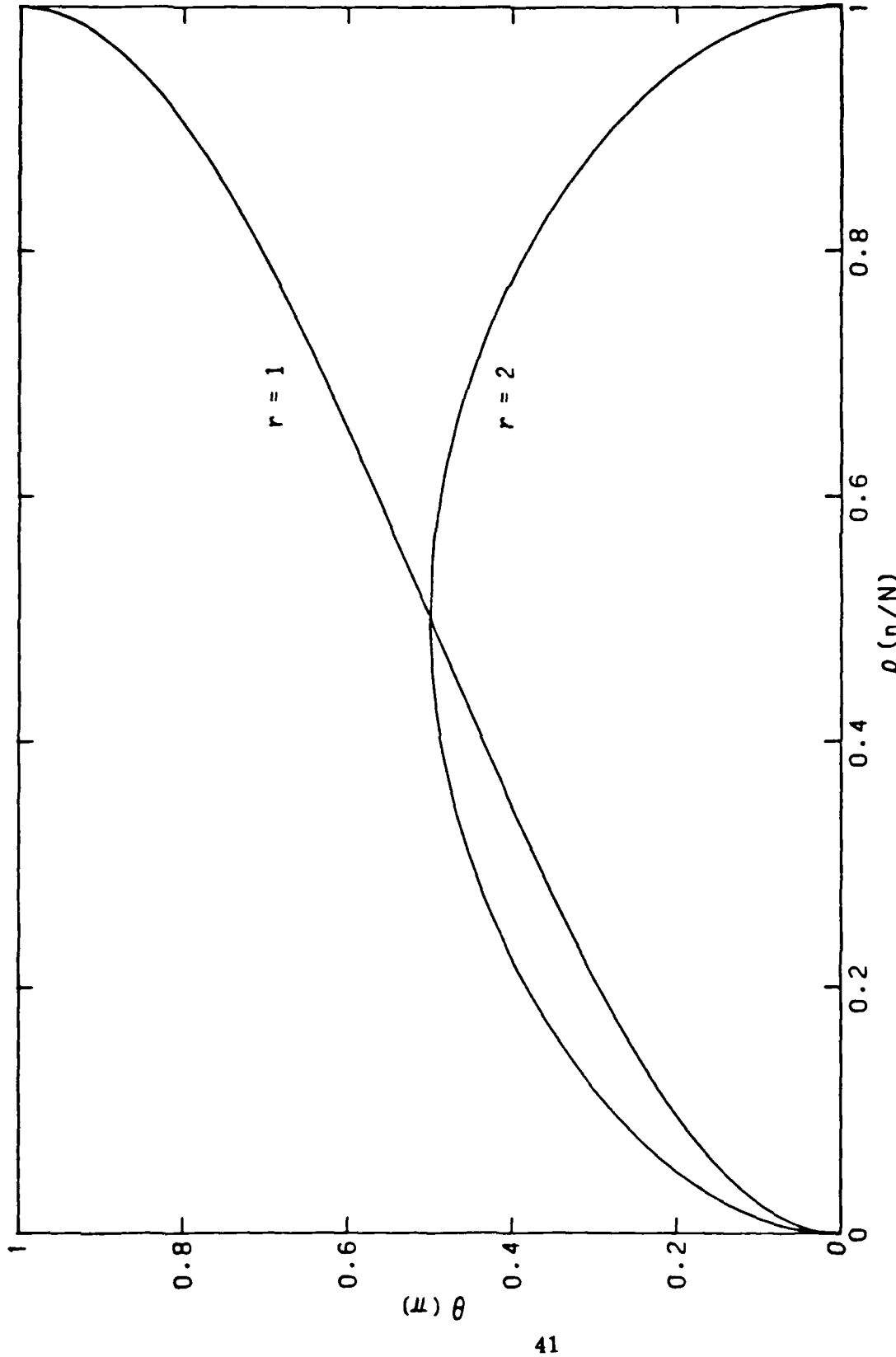


Figure 3a.

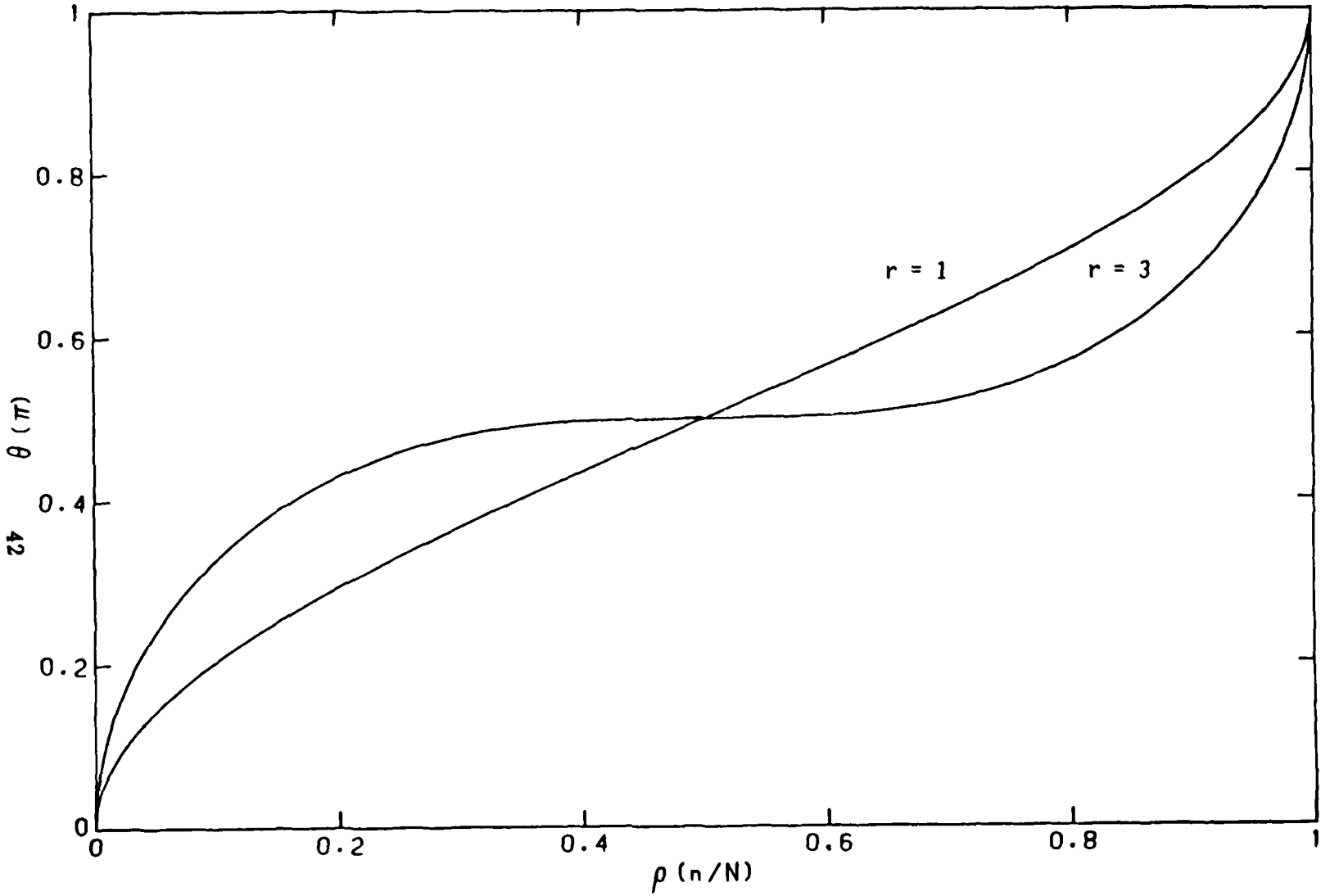


Figure 3b.

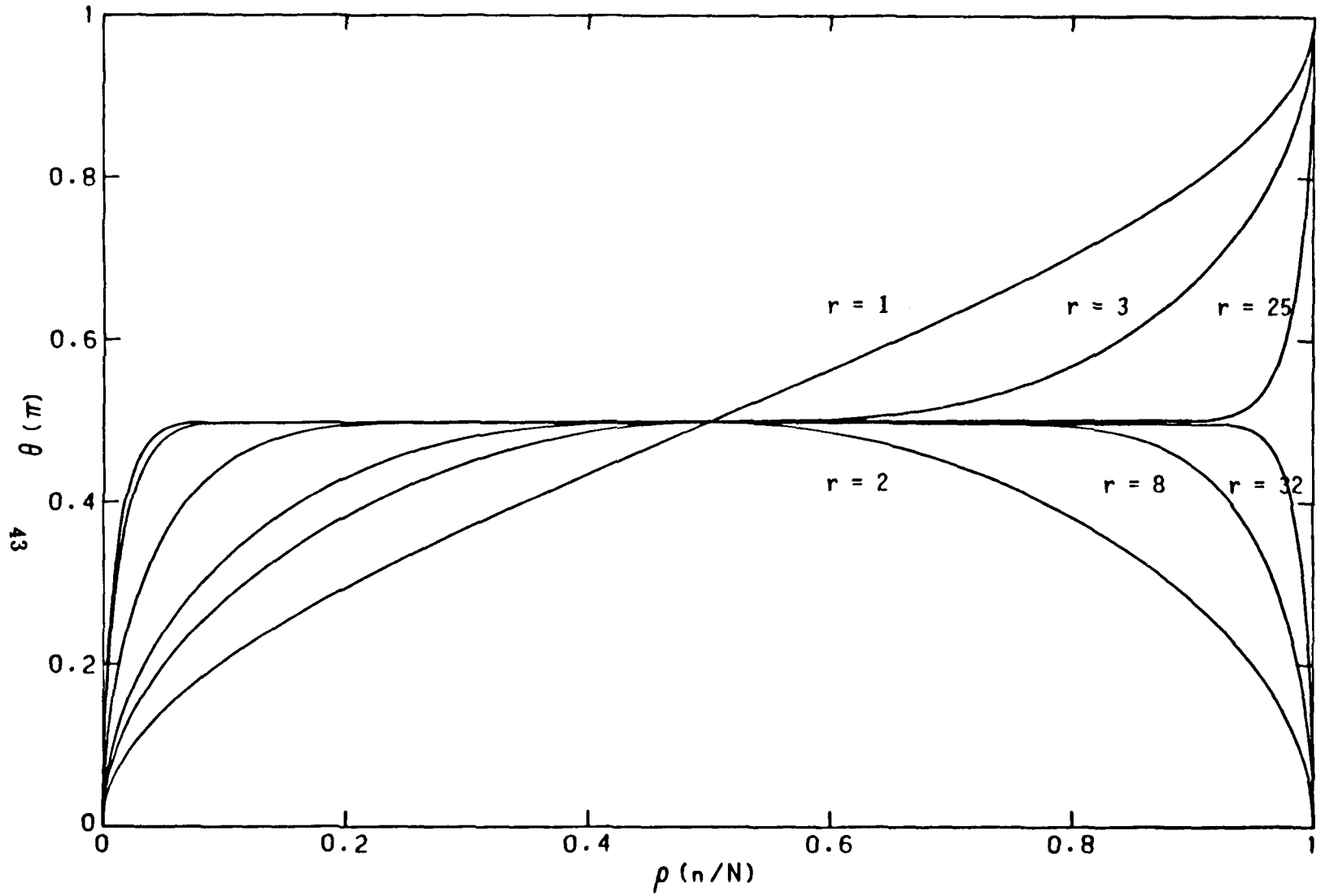


Figure 4.

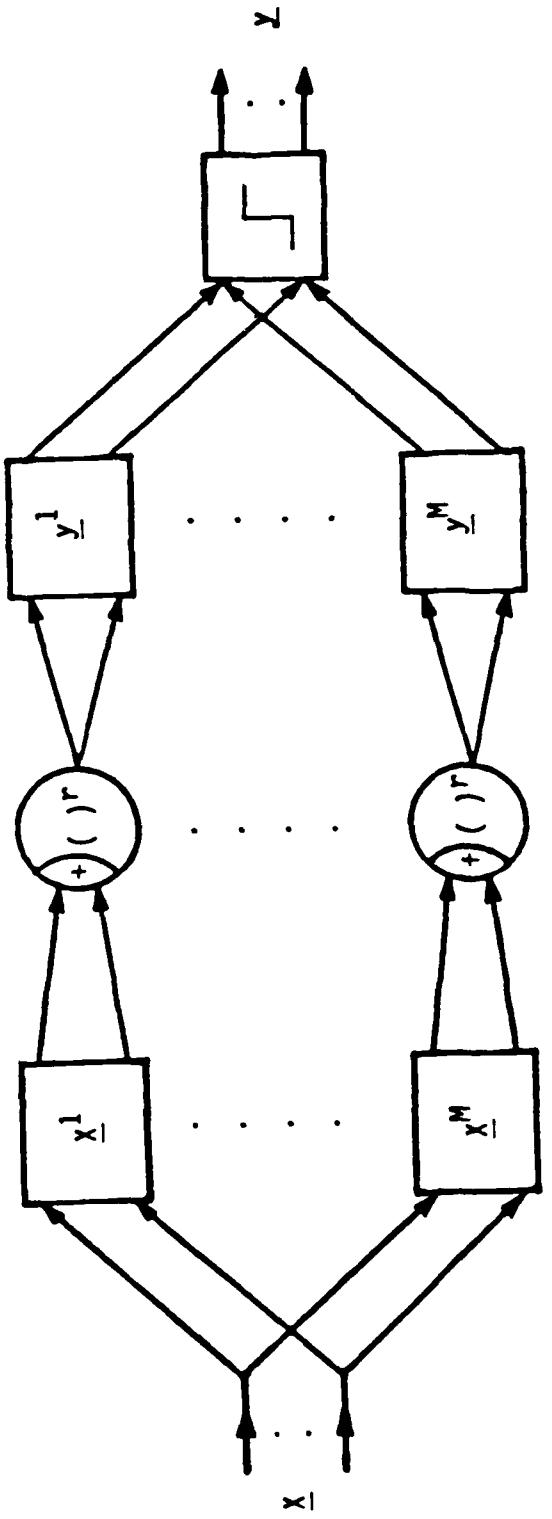
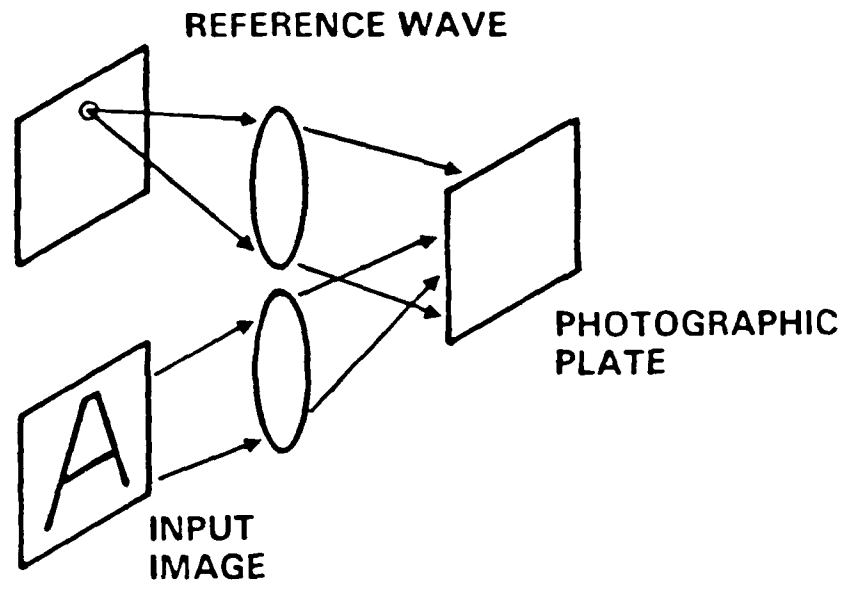
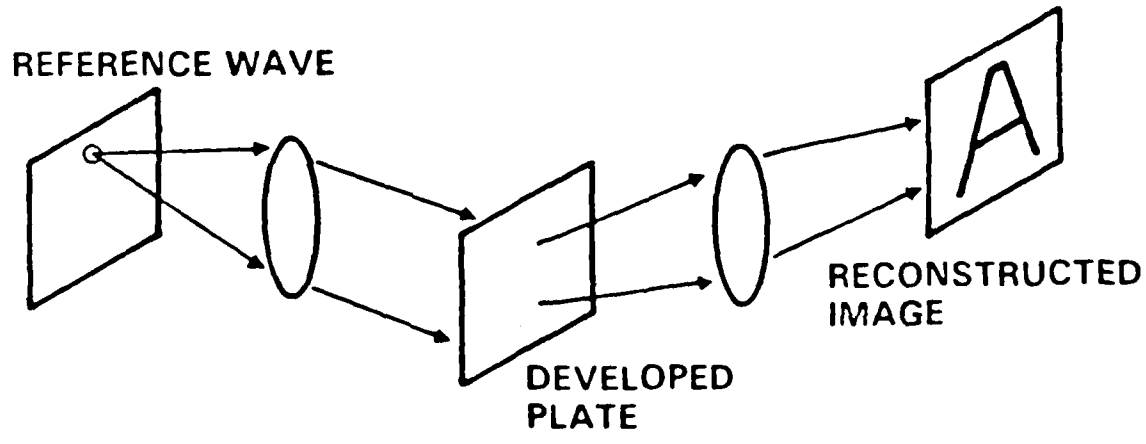


Figure 5.

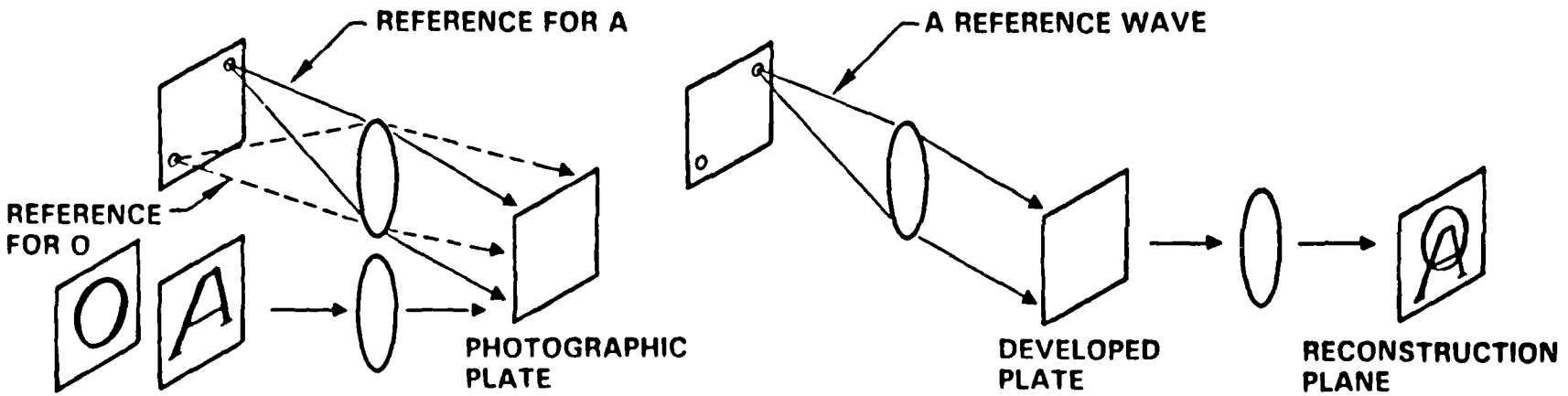


a) RECORDING

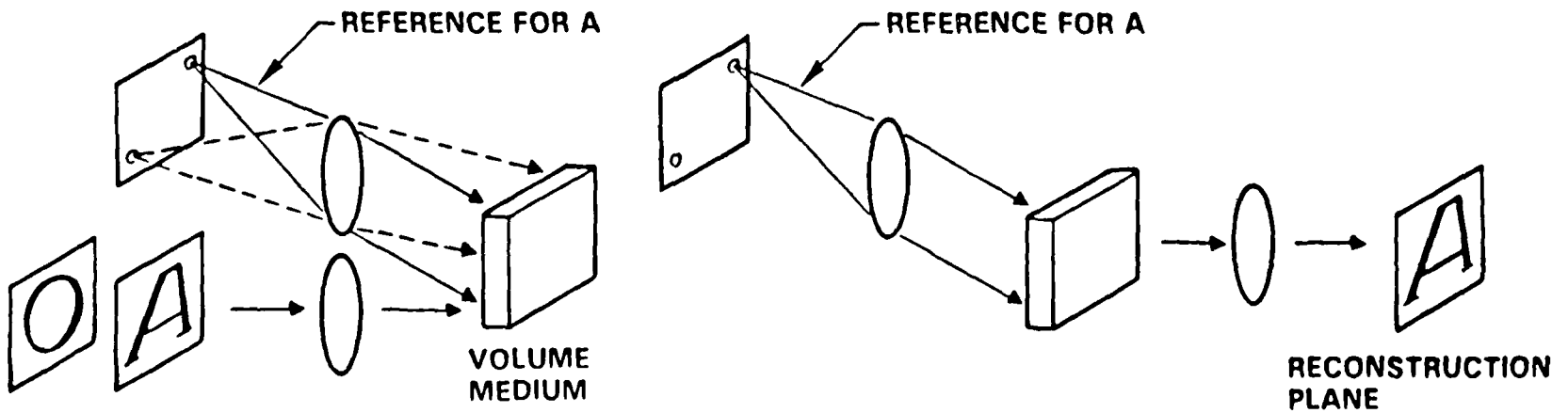


b) RECONSTRUCTION

Figure 6.

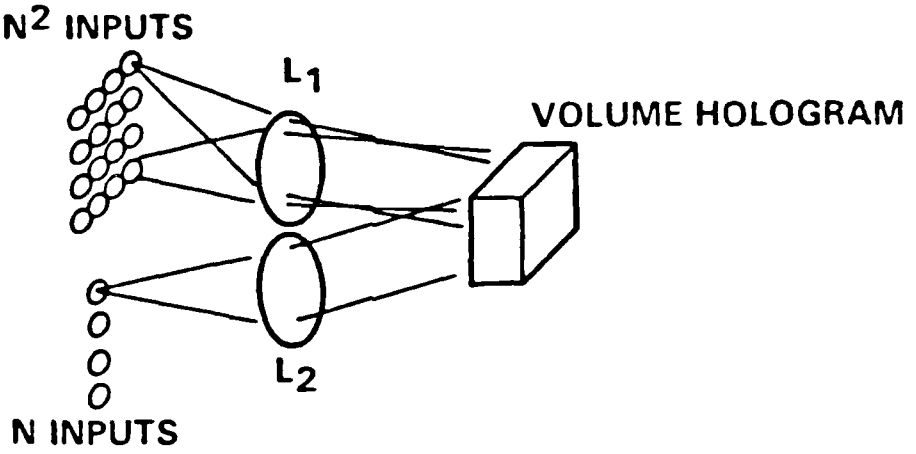


a) PLANAR HOLOGRAM

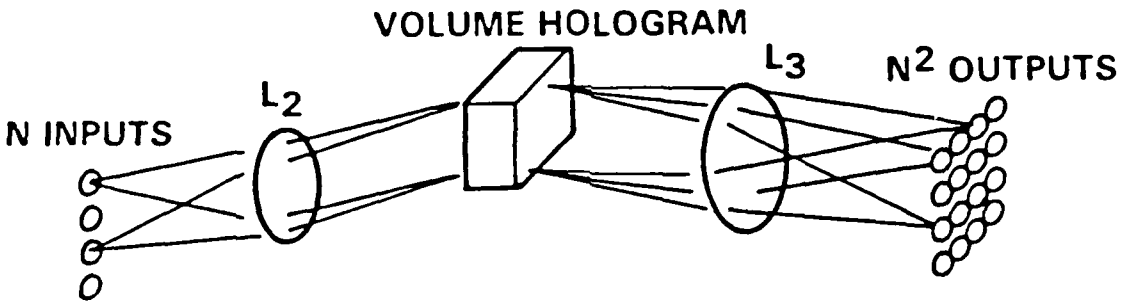


b) VOLUME HOLOGRAM

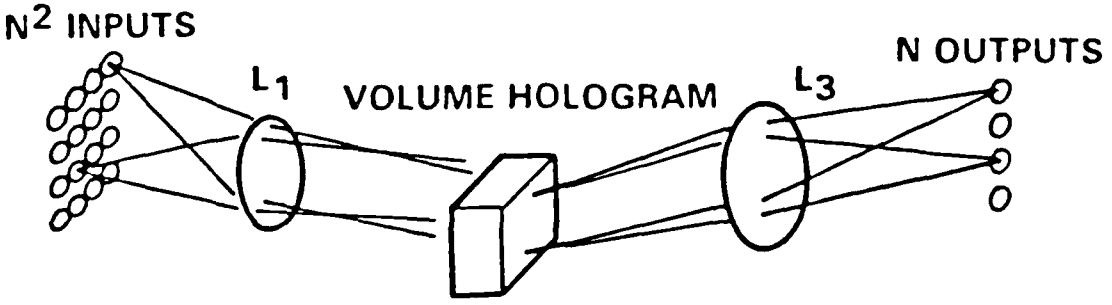
Figure 7.



a) RECORDING



b) N→N2 MAPPING



c) N2→N MAPPING

Figure 8.

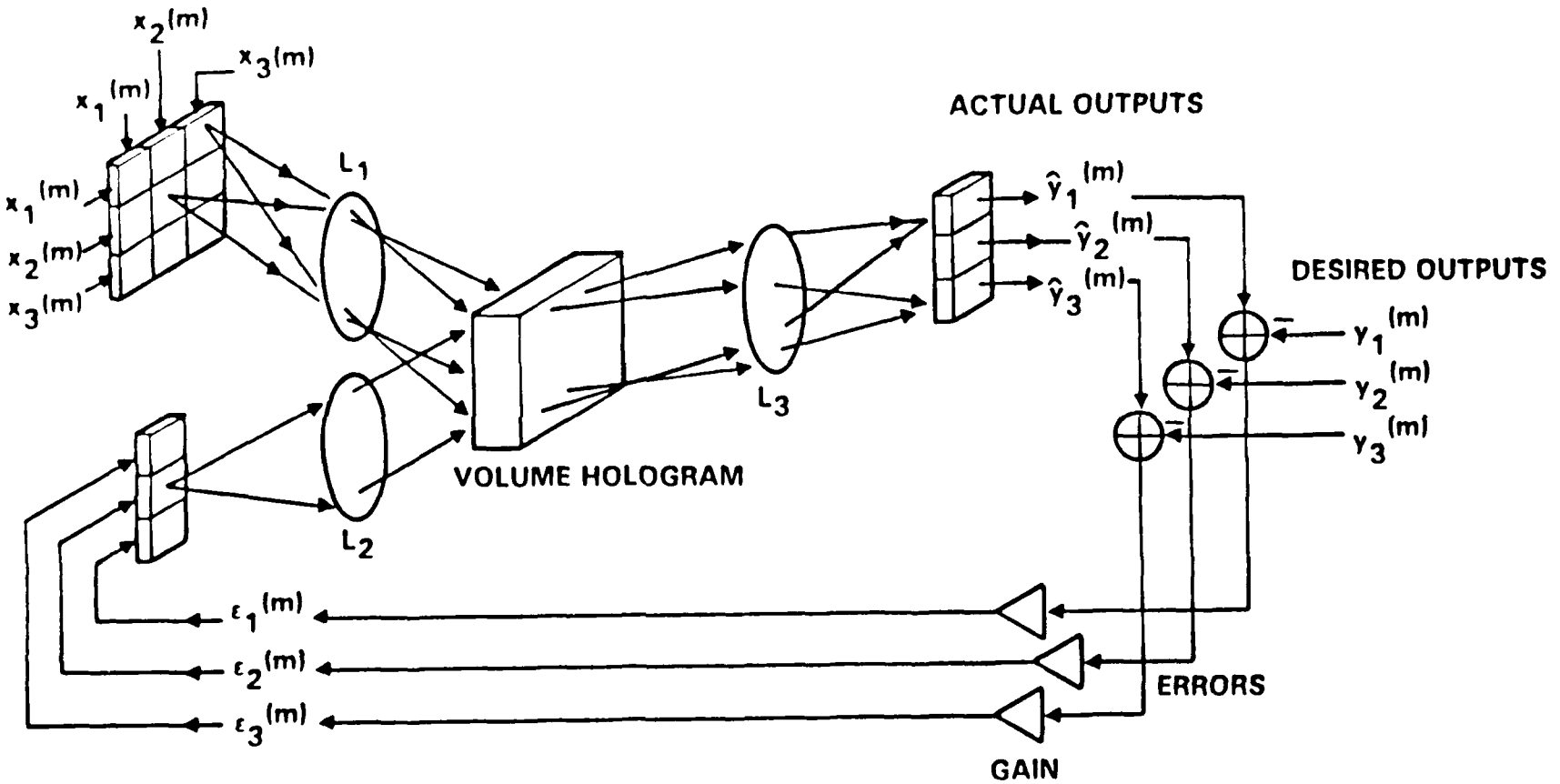

$x_2(m)$
$x_3(m)$
$x_1(m)$
$x_1(m)$
$x_2(m)$
$x_3(m)$
L_1
L_2
L_3
VOLUME HOLOGRAM
ACTUAL OUTPUTS
$\hat{y}_1(m)$
$\hat{y}_2(m)$
$\hat{y}_3(m)$
DESIRED OUTPUTS
$y_1(m)$
$y_2(m)$
$y_3(m)$
$\varepsilon_1(m)$
$\varepsilon_2(m)$
$\varepsilon_3(m)$
ERRORS
GAIN

Figure 9.

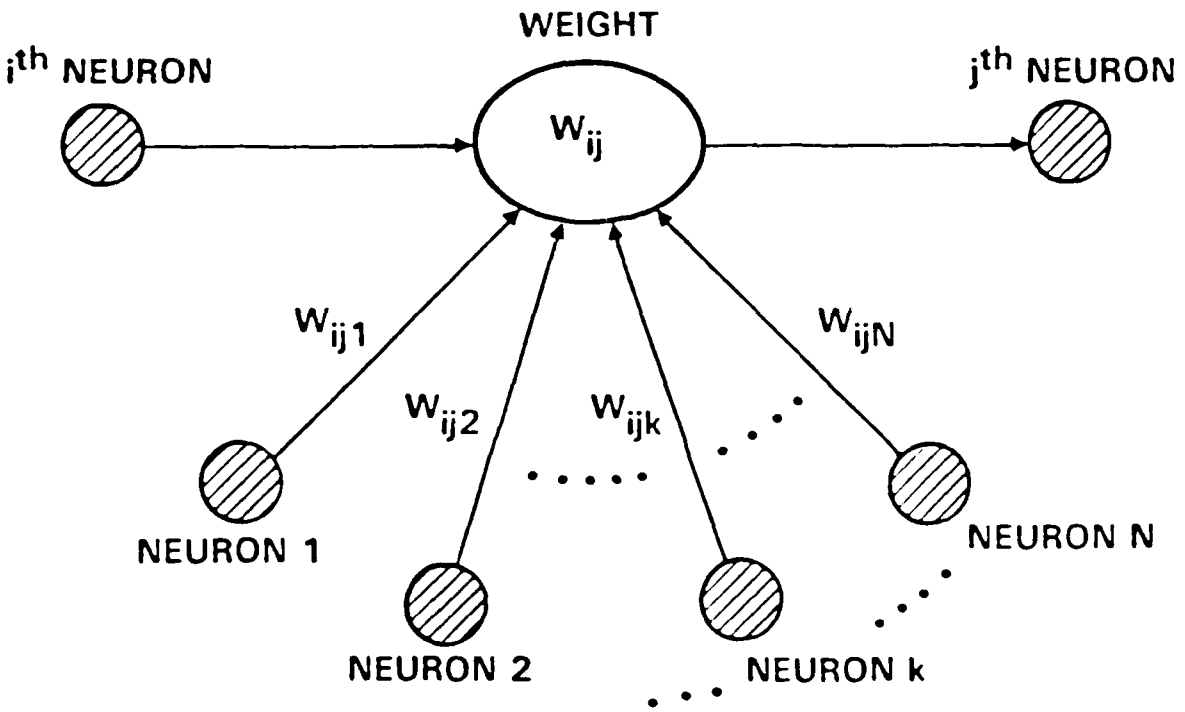

WEIGHT
i^{th} NEURON
j^{th} NEURON
W_{ij}
W_{ij1}
W_{ij2}
W_{ijk}
W_{ijN}
NEURON 1
NEURON 2
NEURON k
NEURON N

Figure 10.

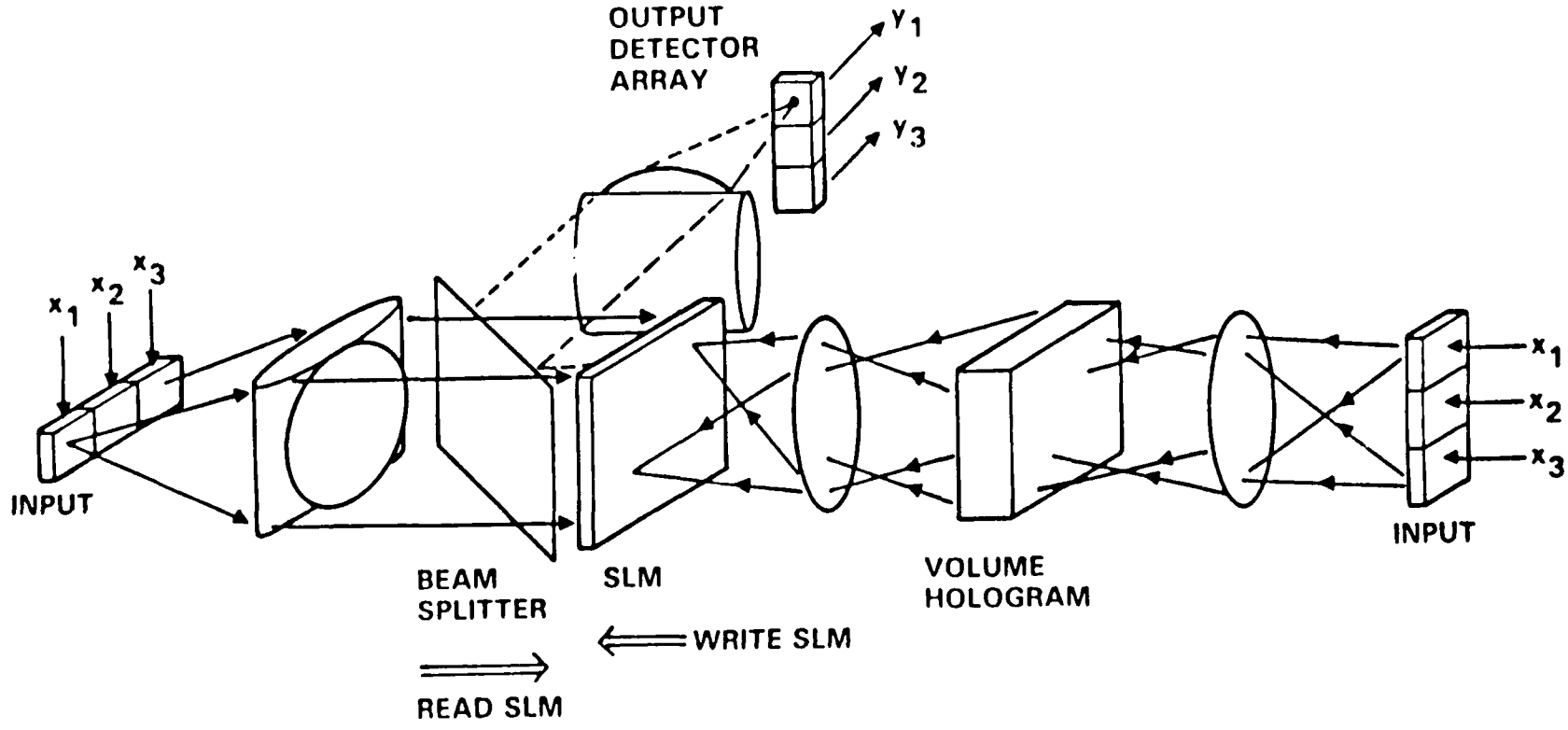

OUTPUT
DETECTOR
ARRAY
y_1
y_2
y_3
x_1
x_2
x_3
INPUT
BEAM
SPLITTER
SLM
WRITE SLM
READ SLM
VOLUME
HOLOGRAM
INPUT
x_1
x_2
x_3

Figure 11.

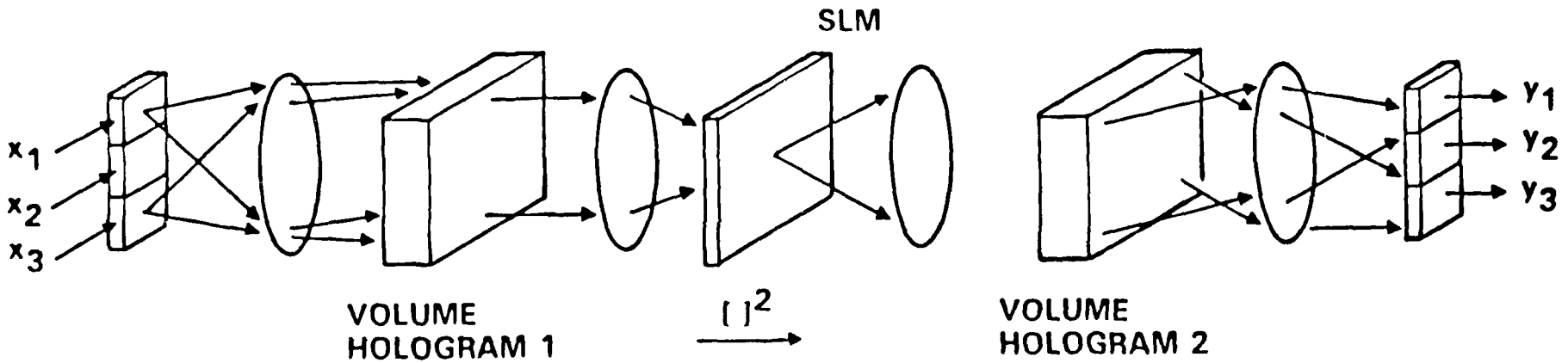


Figure 12.

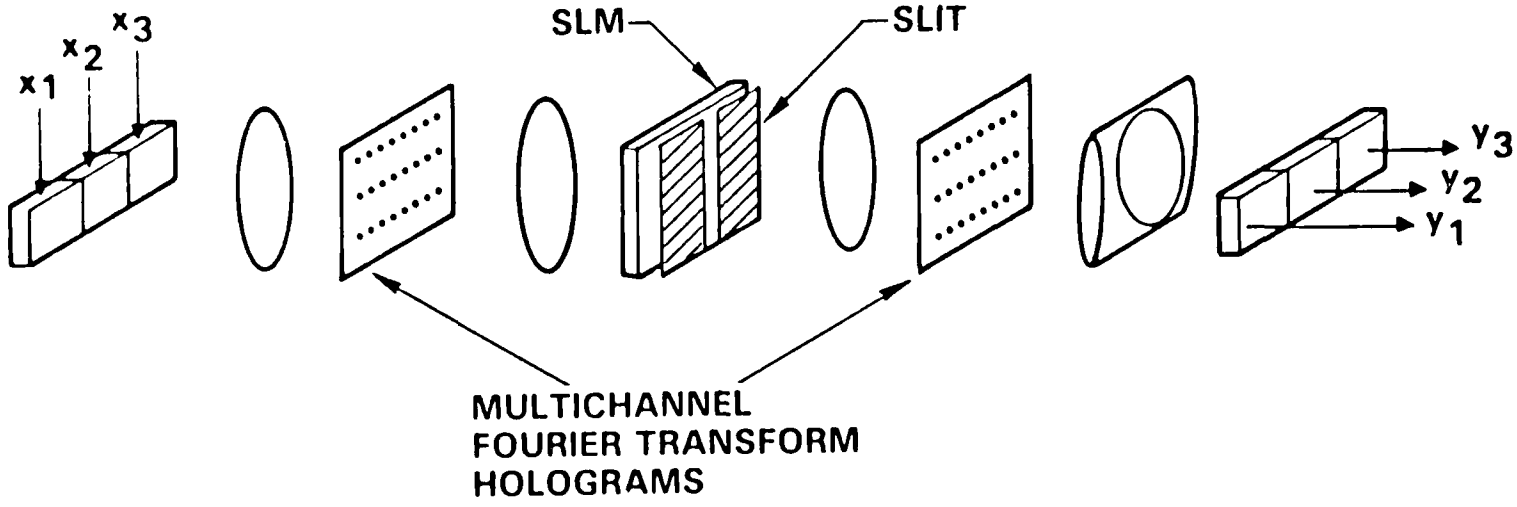

x1
x2
x3
SLM
SLIT
y3
y2
y1
MULTICHANNEL
FOURIER TRANSFORM
HOLOGRAMS

Figure 13.

3 Connectivity versus Entropy

How does the connectivity of a neural network (number of synapses per neuron) relate to the complexity of the problems it can handle (measured by the entropy)? Switching theory would suggest no relation at all, since all Boolean functions can be implemented using a circuit with very low connectivity (e.g., using two-input NAND gates). However, for a network that learns a problem from examples using a *local* learning rule, we prove that the entropy of the problem becomes a lower bound for the connectivity of the network.

3.1 Introduction

The most distinguishing feature of neural networks is their ability to spontaneously learn the desired function from 'training' samples, i.e., their ability to program themselves. Clearly, a given neural network cannot just learn any function, there must be some restrictions on which networks can learn which functions. One obvious restriction, which is independent of the learning aspect, is that the network must be big enough to accommodate the circuit complexity of the function it will eventually simulate. Are there restrictions that arise merely from the fact that the network is expected to *learn* the function, rather than being purposely designed for the function? This paper reports a restriction of this kind.

The result imposes a lower bound on the connectivity of the network (number of synapses per neuron). This lower bound can only be a consequence of the learning aspect, since switching theory provides purposely designed circuits of low connectivity (e.g., using only two-input NAND gates) capable of implementing any Boolean function[1,2]. It also follows that the learning mechanism must be restricted for this lower bound to hold; a powerful mechanism can be designed that will find one of the low-connectivity circuits (perhaps by exhaustive search), and hence the lower bound on connectivity cannot hold in general. Indeed, we restrict the learning mechanism to be local; when a training sample is loaded into the network, each neuron has access only to those bits carried by itself and the neurons it is directly connected to. This is a strong assumption that

excludes sophisticated learning mechanisms used in neural-network models, but may be more plausible from a biological point of view.

The lower bound on the connectivity of the network is given in terms of the *entropy* of the environment that provides the training samples. Entropy is a quantitative measure of the disorder or randomness in an environment or, equivalently, the amount of information needed to specify the environment. There are many different ways to define entropy, and many technical variations of this concept[3]. In the next section, we shall introduce the formal definitions and results, but we start here with an informal exposition of the ideas involved.

The environment in our model produces patterns represented by N bits $\mathbf{x} = x_1 \cdots x_N$ (pixels in the picture of a visual scene if you will). Only h different patterns can be generated by a given environment, where $h < 2^N$ (the entropy is essentially $\log_2 h$). No knowledge is assumed about which patterns the environment is likely to generate, only that there are h of them. In the learning process, a huge number of sample patterns are generated at random from the environment and input to the network, one bit per neuron. The network uses this information to set its internal parameters and gradually tune itself to this particular environment. Because of the network architecture, each neuron knows only its own bit and (at best) the bits of the neurons it is directly connected to by a synapse. Hence, the learning rules are local: a neuron does not have the benefit of the entire global pattern that is being learned.

After the learning process has taken place, each neuron is ready to perform a function *defined by what it has learned.* The collective interaction of the functions of the neurons is what defines the overall function of the network. The main result of this paper is that (roughly speaking) if the connectivity of the network is less than the entropy of the environment, the network cannot learn about the environment. The idea of the proof is to show that if the connectivity is small, the final function of each neuron is independent of the environment, and hence to conclude that the overall network has accumulated no information about the environment it is supposed to learn about.

3.2 Formal result

A neural network is an undirected graph (the vertices are the neurons and the edges are the synapses). Label the neurons $1, \cdots, N$ and define $K_n \subseteq \{1, \cdots, N\}$ to be the set of neurons connected by a synapse to neuron n, together with neuron n itself. An environment is a subset $e \subseteq \{0,1\}^N$ (each $\mathbf{x} \in e$ is a sample from the environment). During learning, $x_1, \cdots, x_N$ (the bits of $\mathbf{x}$) are loaded into the neurons $1, \cdots, N$, respectively. Consider an arbitrary neuron n and relabel everything to make K_n become $\{1, \cdots, K\}$. Thus the neuron sees the first K coordinates of each $\mathbf{x}$.

Since our result is asymptotic in N, we will specify K as a function of N; $K = \alpha N$ where $\alpha = \alpha(N)$ satifies $\lim_{N\to\infty} \alpha(N) = \alpha_o$ $(0 < \alpha_o < 1)$. Since the result is also statistical, we will consider the ensemble of environments $\mathcal{E}$

$$\mathcal{E} = \mathcal{E}(N) = \left\{ e \subseteq \{0,1\}^N \;\mid\; |e| = h \right\}$$

where $h = 2^{\beta N}$ and $\beta = \beta(N)$ satifies $\lim_{N\to\infty} \beta(N) = \beta_o$ $(0 < \beta_o < 1)$. The probability distribution on $\mathcal{E}$ is uniform; any environment $e \in \mathcal{E}$ is as likely to occur as any other.

The neuron sees only the first K coordinates of each $\mathbf{x}$ generated by the environment e. For each e, we define the function $n : \{0,1\}^K \mapsto \{0,1,2,\cdots\}$ where

$$n(a_1 \cdots a_K) = |\{\mathbf{x} \in e \;\mid\; x_k = a_k \text{ for } k = 1, \cdots, K\}|$$

and the normalized version

$$\nu(a_1 \cdots a_K) = \frac{n(a_1 \cdots a_K)}{h}$$

The function ν describes the relative frequency of occurrence for each of the 2^K binary vectors $x_1 \cdots x_K$ as $\mathbf{x} = x_1 \cdots x_N$ runs through all h vectors in e. In other words, ν specifies the projection of e as seen by the neuron. Clearly, $\nu(\mathbf{a}) \geq 0$ for all $\mathbf{a} \in \{0,1\}^K$ and $\sum_{\mathbf{a}\in\{0,1\}^K} \nu(\mathbf{a}) = 1$.

Corresponding to two environments e_1 and e_2, we will have two functions ν_1 and ν_2. If ν_1 is not distinguishable from ν_2, the neuron cannot tell the difference between e_1 and e_2. The distinguishability between ν_1 and ν_2 can

be measured by

$$d(\nu_1, \nu_2) = \frac{1}{2} \sum_{\mathbf{a} \in \{0,1\}^K} |\nu_1(\mathbf{a}) - \nu_2(\mathbf{a})|$$

The range of $d(\nu_1, \nu_2)$ is $0 \leq d(\nu_1, \nu_2) \leq 1$, where '0' corresponds to complete indistinguishability while '1' corresponds to maximum distinguishability. We are now in a position to state the main result.

Let e_1 and e_2 be independently selected environments from $\mathcal{E}$ according to the uniform probability distribution. $d(\nu_1, \nu_2)$ is now a random variable, and we are interested in the expected value $E(d(\nu_1, \nu_2))$. The case where $E(d(\nu_1, \nu_2)) = 0$ corresponds to the neuron getting no information about the environment, while the case where $E(d(\nu_1, \nu_2)) = 1$ corresponds to the neuron getting maximum information. The theorem predicts, in the limit, one of these extremes depending on how the connectivity (α_o) compares to the entropy (β_o).

Theorem.

1. If $\alpha_o > \beta_o$, then $\lim_{N \to \infty} E\left(d(\nu_1, \nu_2)\right) = 1$.

2. If $\alpha_o < \beta_o$, then $\lim_{N \to \infty} E\left(d(\nu_1, \nu_2)\right) = 0$.

The proof is given in the appendix, but the idea is easy to illustrate informally. Suppose $h = 2^{K+10}$ (corresponding to part 2 of the theorem). For most environments $e \in \mathcal{E}$, the first K bits of $\mathbf{x} \in e$ go through all 2^K possible values approximately 2^{10} times each as $\mathbf{x}$ goes through all h possible values once. Therefore, the patterns seen by the neuron are drawn from the fixed ensemble of all binary vectors of length K with essentially uniform probability distribution, i.e., ν is the same for most environments. This means that, statistically, the neuron will end up doing the same function regardless of the environment at hand.

What about the opposite case, where $h = 2^{K-10}$ (corresponding to part 1 of the theorem)? Now, with only 2^{K-10} patterns available from the environment, the first K bits of $\mathbf{x}$ can assume at most 2^{K-10} values out of the possible 2^K values a binary vector of length K can assume in principle. Furthermore, which values can be assumed depends on the particular environment at hand, i.e., ν does depend on the environment. Therefore, although the neuron still does not have the global picture, the information it has says something about the environment.

APPENDIX

In this appendix we prove the main theorem. We start by discussing some basic properties about the ensemble of environments $\mathcal{E}$. Since the probability distribution on $\mathcal{E}$ is uniform and since $|\mathcal{E}| = \binom{2^N}{h}$, we have

$$\Pr(e) = \binom{2^N}{h}^{-1}$$

which is equivalent to generating e by choosing h elements $\mathbf{x} \in \{0,1\}^N$ with uniform probability (without replacement). It follows that

$$\Pr(\mathbf{x} \in e) = \frac{h}{2^N}$$

while for $\mathbf{x}_1 \neq \mathbf{x}_2$,

$$\Pr(\mathbf{x}_1 \in e\ ,\ \mathbf{x}_2 \in e) = \frac{h}{2^N} \times \frac{h-1}{2^N - 1}$$

and so on.

The functions n and ν are defined on K-bit vectors. The statistics of $n(\mathbf{a})$ (a random variable for fixed $\mathbf{a}$) is independent of $\mathbf{a}$

$$\Pr(n(\mathbf{a}_1) = m) = \Pr(n(\mathbf{a}_2) = m)$$

which follows from the symmetry with respect to each bit of $\mathbf{a}$. The same holds for the statistics of $\nu(\mathbf{a})$. The expected value $E(n(\mathbf{a})) = h2^{-K}$ (h objects going into 2^K cells), hence $E(\nu(\mathbf{a})) = 2^{-K}$. We now restate and prove the theorem.

Theorem.

1. If $\alpha_o > \beta_o$, then $\lim_{N\to\infty} E\left(d(\nu_1, \nu_2)\right) = 1$.

2. If $\alpha_o < \beta_o$, then $\lim_{N\to\infty} E\left(d(\nu_1, \nu_2)\right) = 0$.

Proof.

We expand $E\left(d(\nu_1, \nu_2)\right)$ as follows

$$E\left(d(\nu_1, \nu_2)\right) \quad = \quad E\left(\frac{1}{2} \sum_{\mathbf{a}\in\{0,1\}^K} |\nu_1(\mathbf{a}) - \nu_2(\mathbf{a})|\right)$$

$$= \frac{1}{2h} \sum_{\mathbf{a} \in \{0,1\}^K} E\left(|n_1(\mathbf{a}) - n_2(\mathbf{a})|\right)$$
$$= \frac{2^K}{2h} E(|n_1 - n_2|)$$

where n_1 and n_2 denote $n_1(0\cdots0)$ and $n_2(0\cdots0)$, respectively, and the last step follows from the fact that the statistics of $n_1(\mathbf{a})$ and $n_2(\mathbf{a})$ is independent of $\mathbf{a}$. Therefore, to prove the theorem, we evaluate $E(|n_1 - n_2|)$ for large N.

1. Assume $\alpha_o > \beta_o$. Let n denote $n(0\cdots0)$, and consider $\Pr(n = 0)$. For n to be zero, all 2^{N-K} strings $\mathbf{x}$ of N bits starting with K 0's must *not* be in the environment e. Hence

$$\Pr(n = 0) = (1 - \frac{h}{2^N})(1 - \frac{h}{2^N - 1}) \cdots (1 - \frac{h}{2^N - 2^{N-K} + 1})$$

where the first term is the probability that $0\cdots00 \notin e$, the second term is the probability that $0\cdots01 \notin e$ given that $0\cdots00 \notin e$, and so on.

$$\geq \left(1 - \frac{h}{2^N - 2^{N-K}}\right)^{2^{N-K}}$$
$$= \left(1 - h2^{-N}(1 - 2^{-K})^{-1}\right)^{2^{N-K}}$$
$$\geq (1 - 2h2^{-N})^{2^{N-K}}$$
$$\geq 1 - 2h2^{-N}2^{N-K}$$
$$= 1 - 2h2^{-K}$$

Hence, $\Pr(n_1 = 0) = \Pr(n_2 = 0) = \Pr(n = 0) \geq 1 - 2h2^{-K}$. However, $E(n_1) = E(n_2) = h2^{-K}$. Therefore,

$$E(|n_1 - n_2|) = \sum_{i=0}^{h} \sum_{j=0}^{h} \Pr(n_1 = i, n_2 = j)|i - j|$$
$$= \sum_{i=0}^{h} \sum_{j=0}^{h} \Pr(n_1 = i) \Pr(n_2 = j)|i - j|$$
$$\geq \sum_{j=0}^{h} \Pr(n_1 = 0) \Pr(n_2 = j) j$$

$$+ \sum_{i=0}^{h} \Pr(n_1 = i) \Pr(n_2 = 0) i$$

which follows by throwing away all the terms where neither i nor j is zero (the term where both i an j are zero appears twice for convenience, but this term is zero anyway).

$$\begin{aligned} &= \Pr(n_1 = 0) E(n_2) + \Pr(n_2 = 0) E(n_1) \\ &\geq 2(1 - 2h2^{-K}) h2^{-K} \end{aligned}$$

Substituting this estimate in the expression for $E(d(\nu_1, \nu_2))$, we get

$$\begin{aligned} E(d(\nu_1, \nu_2)) &= \frac{2^K}{2h} E(|n_1 - n_2|) \\ &\geq \frac{2^K}{2h} \times 2(1 - 2h2^{-K}) h2^{-K} \\ &= 1 - 2h2^{-K} \\ &= 1 - 2 \times 2^{(\beta - \alpha)N} \end{aligned}$$

Since $\alpha_o > \beta_o$ by assumption, this lower bound goes to 1 as N goes to infinity. Since 1 is also an upper bound for $d(\nu_1, \nu_2)$ (and hence an upper bound for the expected value $E(d(\nu_1, \nu_2))$), $\lim_{N \to \infty} E(d(\nu_1, \nu_2))$ must be 1.

2. Assume $\alpha_o < \beta_o$. Consider

$$\begin{aligned} E(|n_1 - n_2|) &= E\left(|(n_1 - h2^{-K}) - (n_2 - h2^{-K})|\right) \\ &\leq E(|n_1 - h2^{-K}| + |n_2 - h2^{-K}|) \\ &= E(|n_1 - h2^{-K}|) + E(|n_2 - h2^{-K}|) \\ &= 2E(|n - h2^{-K}|) \end{aligned}$$

To evaluate $E(|n - h2^{-K}|)$, we estimate the variance of n and use the fact that $E(|n - h2^{-K}|) \leq \sqrt{\mathrm{var}(n)}$ (recall that $h2^{-K} = E(n)$). Since $\mathrm{var}(n) = E(n^2) - (E(n))^2$, we need an estimate for $E(n^2)$. We write $n = \sum_{\mathtt{a} \in \{0,1\}^{N-K}} \delta_{\mathtt{a}}$, where

$$\delta_{\mathtt{a}} = \begin{cases} 1, & \text{if } 0 \cdots 0\mathtt{a} \in e; \\ 0, & \text{otherwise.} \end{cases}$$

In this notation, $E(n^2)$ can be written as

$$\begin{aligned} E(n^2) &= E\left(\sum_{\mathbf{a}\in\{0,1\}^{N-K}} \sum_{\mathbf{b}\in\{0,1\}^{N-K}} \delta_{\mathbf{a}}\delta_{\mathbf{b}}\right) \\ &= \sum_{\mathbf{a}\in\{0,1\}^{N-K}} \sum_{\mathbf{b}\in\{0,1\}^{N-K}} E(\delta_{\mathbf{a}}\delta_{\mathbf{b}}) \end{aligned}$$

For the 'diagonal' terms ($\mathbf{a} = \mathbf{b}$),

$$\begin{aligned} E(\delta_{\mathbf{a}}\delta_{\mathbf{a}}) &= \Pr(\delta_{\mathbf{a}} = 1) \\ &= h2^{-N} \end{aligned}$$

There are 2^{N-K} such diagonal terms, hence a total contribution of $2^{N-K} \times h2^{-N} = h2^{-K}$ to the sum. For the 'off-diagonal' terms ($\mathbf{a} \neq \mathbf{b}$),

$$\begin{aligned} E(\delta_{\mathbf{a}}\delta_{\mathbf{b}}) &= \Pr(\delta_{\mathbf{a}} = 1, \delta_{\mathbf{b}} = 1) \\ &= \Pr(\delta_{\mathbf{a}} = 1)\Pr(\delta_{\mathbf{b}} = 1|\delta_{\mathbf{a}} = 1) \\ &= \frac{h}{2^N} \times \frac{h-1}{2^N - 1} \end{aligned}$$

There are $2^{N-K}(2^{N-K} - 1)$ such off-diagonal terms, hence a total contribution of $2^{N-K}(2^{N-K} - 1) \times \frac{h(h-1)}{2^N(2^N-1)} \leq (h2^{-K})^2 \frac{2^N}{2^N-1}$ to the sum. Putting the contributions from the diagonal and off-diagonal terms together, we get

$$\begin{aligned} E(n^2) &\leq h2^{-K} + (h2^{-K})^2 \frac{2^N}{2^N - 1} \\ \mathrm{var}(n) &= E(n^2) - (E(n))^2 \\ &\leq \left(h2^{-K} + (h2^{-K})^2 \frac{2^N}{2^N - 1}\right) - \left(h2^{-K}\right)^2 \\ &= h2^{-K} + (h2^{-K})^2 \frac{1}{2^N - 1} \\ &= h2^{-K}\left(1 + \frac{h2^{-K}}{2^N - 1}\right) \\ &\leq 2h2^{-K} \end{aligned}$$

The last step follows since $h2^{-K}$ is much smaller than $2^N - 1$. Therefore, $E(|n - h2^{-K}|) \leq \sqrt{\mathrm{var}(n)} \leq \left(2h2^{-K}\right)^{\frac{1}{2}}$. Substituting this estimate in the

expression for $E(d(\nu_1, \nu_2))$, we get

$$\begin{aligned} E(d(\nu_1,\nu_2)) &= \frac{2^K}{2h} E(|n_1 - n_2|) \\ &\leq \frac{2^K}{2h} \times 2E(|n - h2^{-K}|) \\ &\leq \frac{2^K}{2h} \times 2 \times \left(2h2^{-K}\right)^{\frac{1}{2}} \\ &= \left(2\frac{2^K}{h}\right)^{\frac{1}{2}} \\ &= \sqrt{2} \times 2^{\frac{1}{2}(\alpha-\beta)N} \end{aligned}$$

Since $\alpha_o < \beta_o$ by assumption, this upper bound goes to 0 as N goes to infinity. Since 0 is also a lower bound for $d(\nu_1, \nu_2)$ (and hence a lower bound for the expected value $E(d(\nu_1, \nu_2))$), $\lim_{N\to\infty} E(d(\nu_1, \nu_2))$ must be 0.

References

[1] Y. Abu-Mostafa, "Neural networks for computing?," *AIP Conference Proceedings # 151, Neural Networks for Computing*, J. Denker (ed.), pp. 1-6, 1986.

[2] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill, 1978.

[3] Y. Abu-Mostafa, "The complexity of information extraction," *IEEE Trans. on Information Theory*, vol. IT-32, pp. 513-525, July 1986.

[4] Y. Abu-Mostafa, "Complexity in neural systems," in *Analog VLSI and Neural Systems* by C. Mead, Addison-Wesley, 1988.

4 Adaptive Optical Networks Using Photorefractive Crystals

4.1 Introduction

Learning is the most distinctive feature of a neural computer and in many respects it is this aspect that gives neural computation an advantage over alternative computational strategies. A neural computer is trained to produce the appropriate response to a class of inputs by being presented with a sufficient number of examples during the learning phase. The presentation of these examples causes the strength of the connections between neurons that comprise the network to be modified according to the specifics of the learning algorithm. A successful learning procedure will result in a trained network that responds correctly when it is presented with the examples it has seen previously and also other inputs that are in some sense similar to the known patterns. When we consider a physical realization of a neural network model, we have two options in incorporating learning capability. The first is to build a network with fixed but initially programmable connections. An auxiliary, conventional computer can then be used to "learn" the correct values of the connection strengths and once learning has been completed the network can be programmed by the computer. While this approach may be reasonable for some applications, a system with continuously modifiable connections presents a much more powerful alternative.

In this paper we consider the optical implementation of learning networks using volume holographic interconnections in photorefractive crystals. The use of volume holograms permits the storage of a very large number of interconnections per unit volume [1,2,3,4] whereas the use of photorefractive crystals permits the dynamic modification of these connections, thus allowing the implementation of learning algorithms [5,6,7,8,9]. We first briefly review the major types of learning algorithms that are being used in neural network models. We then estimate the maximum number of holographic gratings that can simultaneously exist in a photorefractive crystal. Since in an optical implementation each grating corresponds to a separate interconnection between two neurons, this estimate gives us the density of connections that are achievable with volume holograms. The

next topic that we address is how the modulation depth of each grating (or equivalently the strength of each connection) can be controlled through the implementation of learning algorithms. Two related issues are investigated: the optical architectures which implement different learning algorithms and the reconciliation of physical mechanisms that are involved in the recording of holograms in photorefractive crystals with the dynamics of the learning procedures in neural networks.

4.2 Learning algorithms

For the purposes of this discussion it is convenient to separate the wide range of learning algorithms that have been discussed in the literature into three categories: prescribed learning, error driven learning and self organization. We will draw the distinction among these with the aid of Fig. 1, where a general network is drawn with the vector $\underline{x}(k)$ as its input and $\underline{y}(k)$ the output at the k^{th} iteration (or time interval). The vector $\underline{z}(k)$ is used to represent the activity of the internal units and $w_{ij}(k)$ is the connection strength between the i^{th} and the j^{th} unit. Let $\underline{x}^{(m)}$, $m = 1...M$, be a set of specified input vectors and let $\underline{y}^{(m)}$ be the responses which the network must produce for each of these input vectors.

A prescribed learning algorithm calculates the strength of each weight simply as a function of the vectors $\underline{x}^{(m)}$ and $\underline{y}^{(m)}$:

$$w_{ij} = f_{ij}(\underline{x}^{(m)}, \underline{y}^{(m)}) \qquad m = 1...M \tag{1}$$

This type of procedure is relatively simple ("easy learning"). It is perhaps the most sensible approach in a single layer network. The widely used outer product algorithm [10,11] is an example of this type of learning algorithm, as are some schemes which utilize the pseudoinverse [10,12,13]. Despite its simplicity, prescribed learning is limited in several important respects. First, while prescribed learning is well understood for single layer systems, the existing algorithms for two layers are largely localized representations; each input $\underline{x}^{(m)}$ activates a single internal neuron [14,15,16]. Moreover, the entire learning procedure usually has to be completed a priori. This last limitation is not encountered in the simplest form of prescribed learning,

the outer product rule:

$$w_{ij} = \sum_{m=1}^{M} x_i^{(m)} y_j^{(m)} \tag{2}$$

In this case new memories may be programmed by simply adding the outer products of new samples to the weight matrix. Note that once the interconnection matrix has been determined by a prescribed learning algorithm, it may be expressed in the form of a sum of at most N outer products, where N is the total number of neurons in each layer. Since volume holograms record interconnections matrices represented by sums of outer products in a very natural way, matrices which can be expressed in this form are particularly simple to implement in optics [17,18,19,20].

Error driven learning is distinguished by the fact that the output of the system, $\underline{y}(k)$, is monitored and compared to the desired response $\underline{y}^{(m)}$. An incremental change is then made to the interconnection weights to reduce the error.

$$\Delta w_{ij}(k) = f_{ij}[\underline{x}^{(m)}, w_{rs}(k), \underline{y}^{(m)}] \tag{3}$$

The change Δw_{ij} is calculated from the vectors $\underline{x}^{(m)}$ and $\underline{y}^{(m)}$ and the current setting of the weight matrix $w_{rs}(k)$ (from which the state of the entire network can be calculated). The perceptron [21] and adaline [22] algorithms are examples of error driven learning for single layer networks. Interest in such learning algorithms has been renewed recently by the development of procedures suitable for multi-layered networks [23,24,25]. Error driven algorithms ("hard learning") are more difficult to implement than prescribed learning since they require a large number of iterations before errors can be reduced to sufficiently low levels. In multilayered systems, however, this type of learning can provide an effective mechanism for matching the available resources (connections and neurons) to the requirements of the problem. In optical realizations error driven algorithms are more difficult to implement than prescribed approaches due to the need for dynamically modifiable interconnections and the incorporation of an optical system that monitors the performance and causes the necessary changes in the weights [26]. While this problem could be avoided by performing learning off line in computer simulations and recording the optimized interconnection matrix as in prescribed learning, this approach has the disadvantage that once

again the matrix is fixed a priori, thus preventing the network from being adaptive. In subsequent sections we will consider a relatively simple form of Eqn. (3) in which $\Delta w_{ij}(k)$ depends only on locally available information, i.e. z_i in one layer and z_j in an adjacent layer

$$\Delta w_{ij}(k) = f_{ij}[z_i\{w_{rs}(k), \underline{y}^{(m)}, \underline{x}^{(m)}\}, z_j\{w_{rs}(k), \underline{y}^{(m)}, \underline{x}^{(m)}\}] \tag{4}$$

The perceptron and the backwards error propagation algorithms both fall in this subcategory if we allow the neuronal activity z_i to include error signals, i.e. if each neuron has distinct signal and error outputs which are separated temporally or spatially. An example of such a neuron implemented in optics is given below in conjunction with an optical back error propagation system.

In the case of self organizing learning algorithms we require not that the specified inputs produce a particular response but rather that they satisfy a general restriction, often imposed by the structure of the network itself. Since there is no a priori expected response, the learning rule for self organizing systems is simply

$$\Delta w_{ij}(k) = f_{ij}[\underline{x}^{(m)}, w_{rs}(k)] \tag{5}$$

This type of learning procedure can be useful, for instance, at intermediate levels of a network where the purpose is not to elicit an external response but rather to generate appropriate internal representations of the information that is presented as input to the network. There is a broad range of self organizing algorithms, the simplest of which is probably lateral inhibition to enforce grandmother cell representations [10,27]. The objective of the learning procedure is to have each distinct pattern in an input set of neurons activate a single neuron in a second set. In the architecture shown in Fig. 2 this is accomplished via inhibitory connections between the neurons in the second set. Once a particular neuron in the second layer is partially turned on for a specific pattern it prevents the connections to the other neurons in the second set from assuming values that will result in activity at more than one neuron. The details of the dynamics of such procedures can be quite complex (e.g. [28]), as can corresponding optical implementations. An advantageous feature of optics in connection with self organization is that global training signals, such as fixed lateral inhibition between all the neurons in a given layer, can easily be broadcast with optical beams.

4.3 Interconnection capabilities of volume holograms

The basic architecture for optical implementation of a neural computer is shown in Fig. 3. The figure presents a single stage of what may be a multilayered system. The nonlinear processing elements (i.e. the "neurons") are arranged in planes. We have included a "training plane" for reasons which will become clear below. Neurons in one plane are interconnected with the neurons in the same or other planes via the third dimension. The strength of the interconnections is determined by the information which is holographically stored in light sensitive media placed in the space separating the neural planes. Volume, rather than "thin", holograms are specified in Fig. 3 due to the much greater storage capacity of a volume holograms and the availability of excellent real time volume media. Photorefractive crystals are particularly attractive as holographic media in this application because it is possible to record information in these crystals in real time at very high density without degrading the photorefractive sensitivity. In this section we discuss the factors that determine the maximum number of connections that can be specified by a photorefractive crystal with a given set of physical characteristics. There are three distinct factors that need to be considered: geometric limitations arising from the basic principles of volume holography, limitations rising from the physics of photorefractive recording, and limitations due to the learning algorithms.

The Fourier lenses in Fig. 3 transform the spatial position of each neuron into a spatial frequency associated with light emitted by or incident on that neuron. An interconnection between the i^{th} neuron in the input plane and the j^{th} neuron in the output plane is formed by interfering light emitted by the input neuron with light emitted by the j^{th} neuron in the training plane. The image of the j^{th} training neuron lies at the position of the j^{th} neuron in the output plane. The interference of the training signal and the input creates a grating in the recording medium of the form

$$\Delta\chi_{ij} = A_i A_j^* e^{j\vec{K}_{ij}\cdot\vec{r}} \tag{6}$$

where A_i and A_j are the amplitudes of the fields emitted by the i^{th} and j^{th} neurons, respectively. $\vec{K}_{ij}$ is equal to $\vec{k}_i - \vec{k}_j$ where $\vec{k}_i$ and $\vec{k}_j$ are the spatial frequencies at which the corresponding amplitudes propagate in the

volume medium. This grating diffracts an input beam at spatial frequency $\vec{k}_\alpha$ into an output beam at spatial frequency $\vec{k}_\beta$ if these two beams satisfy the Bragg constraint that

$$\vec{k}_\alpha - \vec{k}_\beta = \vec{K}_{ij} \tag{7}$$

This constraint is obviously satisfied if $\vec{k}_\alpha = \vec{k}_i$ and $\vec{k}_\beta = \vec{k}_j$. In general this solution is not unique. However, Psaltis *et al.* [2,3] have shown that by placing the neurons on the input and output planes on appropriate fractal grids of dimension $\frac{3}{2}$ it is possible to insure that only the i^{th} input neuron and the j^{th} output neuron may be coupled by a grating with wavevector K_{ij}. In this case, recording a hologram between light from the i^{th} input neuron and the j^{th} training neuron increases the connection strength between the i^{th} input and the j^{th} output without directly affecting the connections between other neurons. If instead of one neuron, patterns of neurons are active on the fractal grids of the input and training planes then the hologram recorded in the volume, i.e. Eqn. (6) summed over all active pairs of neurons, is the outer product of the pattern on the input plane and the pattern on the training plane. Exposing the hologram with a series of M pattern yields the sum of outer products described by Eqn. (2). Note that the architecture shown in Fig. 3 is similar to a joint Fourier transform correlator. The use of volume, rather than thin, holograms and fractal grids destroys the shift invariance of the correlator, making this architecture a totally shift variant arbitrarily interconnectable system.

A basic geometrical limitation on the density of interconnections achievable through volume holograms is due to the finite volume, V, of any real crystal. The refractive index $n(\vec{r})$ of such a crystal under periodic boundary conditions may be represented in the form

$$n(\vec{r}) = \sum_{\nu}^{S} n_\nu e^{j\vec{k}_\nu \cdot \vec{r}} \tag{8}$$

$$\vec{k}_\nu = (\nu_x(\frac{2\pi}{L_x})\hat{x} + \nu_y(\frac{2\pi}{L_y})\hat{y} + \nu_z(\frac{2\pi}{L_z})\hat{z}) \quad \nu_i = 0, \pm 1, \pm 2... \tag{9}$$

Where n_ν is the amplitude of the Fourier component at spatial frequency $\vec{k}_\nu$ and L_i is the length of the crystal in the $\hat{i}$ direction. Since the maximum spatial frequency which may be Bragg matched to diffract light at wavelength λ is $2k_o$, where $k_o = \frac{2\pi}{\lambda}$ the sum in Eqn. 8 is finite in holographic

applications. The number of spatial frequencies in the sum is $S \approx \frac{V}{\lambda^3}$. Psaltis *et al.* [2,3] demonstrated that S is sufficient to fully and independently interconnect neural planes which are limited to fractal dimension $\frac{3}{2}$. Thus in this previous work the issue of these geometric limitations was fully resolved under the condition that processing nodes in the input and output planes must be appropriately arranged on fractal grids. Other geometric limitations arise due to finite numerical apertures and the physics of holographic recording mechanisms. These factors may be shown to contribute a scaling factor to S which is independent of V and λ. For $V = 1$ cm^3 and $\lambda = 1\mu$m, $\frac{V}{\lambda^3}$ is equal to 10^{12}. In interconnecting neurons arranged on fractal planes, even though the recording geometry typically allows access to only 1% of grating wavevector space, we still may achieve 10^{10} interconnections per cm^3.

We now address the question of whether this large number of gratings can be supported in a photorefractive crystal, i.e. do photorefractive crystals have the capability of simultaneously storing 10^{10} gratings each with sufficient diffraction efficiency? In this paper we answer this question based on simple arguments in the context of a neural architecture. The conclusions we reach are the same as those we arrive at through a more thorough examination of the problem. Photorefractive holograms are produced in electrooptic crystal via the modulation of the index of refraction by the space charge field created by an optically driven inhomogeneous charge distribution. A neural network architecture implemented in volume holograms performs a transformation of the form

$$E_{i\,in}e^{j\vec{k}_i\cdot\vec{r}}e^{j\phi_i} + c.c. = \sum_j \eta_{ij}e^{j\psi_{ij}}e^{j\vec{K}_{ij}\cdot\vec{r}}E_{j\,out}e^{j\vec{k}_j\cdot\vec{r}}e^{j\phi_j} + c.c. \qquad (10)$$

between the field amplitude, $E_{j\,out}e^{j\vec{k}_j\cdot\vec{r}}$, of the j^{th} neuron and the field amplitude, $E_{i\,in}e^{j\vec{k}_i\cdot\vec{r}}$, incident on the input of the i^{th} neuron. *c.c.* denotes the complex conjugate of the preceding term. ϕ_j and ϕ_i are the phases of the field amplitudes corresponding to the i^{th} and j^{th} neurons. ψ_{ij} is the phase of the grating which connects the i^{th} and j^{th} neurons. The diffraction efficiencies η_{ij} are proportional to the component of the space charge density in the crystal at spatial frequency $\vec{K}_{ij} = \vec{k}_i - \vec{k}_j$ [29]. The total space charge density due to N stored gratings is constrained at every point in the crystal

to be less than the acceptor trap density. This implies that

$$\Re\{\sum_i \sum_j \eta_{ij} e^{j\psi_{ij}} e^{j\vec{K}_{ij}\cdot\vec{r}}\} \leq \eta_o \tag{11}$$

where η_o is the maximum diffraction efficiency when only one grating is recorded. If ψ_{ij} is an independent uniformly distributed random variable on $(-\pi, \pi)$, then with high probability the right side of Eqn. (11) will not exceed a few times its standard deviation, $\sqrt{\frac{N}{2}}\eta_1$, where η_1 is the rms value of η_{ij}. This fact allows us to find a simple limit for η_1 given by

$$\eta_1 \approx \frac{\eta_o}{\sqrt{\frac{N}{2}}} \tag{12}$$

Note that although we have assumed that the sums in Eqn. (11) are over a set of incoherent sinusoids, this does not imply that the sum in Eqn. (10) is incoherent. To illustrate this point imagine that $\psi_{ij} = \phi_i - \phi_j$. In this case the terms in Eqn. (10) add coherently. However if ϕ_i and ϕ_j are independent random variables the sums in Eqn. (11) still add incoherently. Thus a random phase term in the transmittance at each neuron causes the charge densities stored in the crystal to add incoherently but does not necessarily destroy the coherence of the optical system.

The holographic transformation described above can be used to implement neural architectures which map an activity pattern described by the outputs $\{x_j\}$ of the neurons on one neural plane to the outputs $\{y_i\}$ of the next neural plane. In a coherent optical system x_j is represented by $E_{j\,out}e^{j\phi_j}$ and w_{ij} is represented by $\eta_{ij}e^{j\psi_{ij}}$. Since most simple optical nonlinearities are based on absorption the transformation between $\{x_j\}$ and $\{y_i\}$ typically takes the form

$$y_i = f(|\sum_j w_{ij}x_j|^2) \tag{13}$$

where f is a thresholding function implemented in the neural plane. This functional form might be avoided using interferometic detection. In an incoherent optical system x_j is represented by $|E_{j\,out}|^2$ and w_{ij} is represented by η_{ij}^2. The transformation between $\{x_j\}$ and $\{y_i\}$ takes the form

$$y_i = f(\sum_j w_{ij}x_j) \tag{14}$$

In either case the function f must provide sufficient gain, G, to regenerate the signal power of the system after each layer. If we assume that each layer contains $\sqrt{N}$ neurons then the relationship between the power incident on a single neuron, I_{in}, and the power output by a single neuron, I_{out}, for a coherent system with $\psi_{ij} = \phi_i - \phi_j$ is

$$I_{in} = \kappa |\sum_{j}^{\sqrt{N}} \eta_{ij} e^{j\psi_{ij}} E_{j\ out} e^{j\phi_j}|^2 = N\eta_1^2 I_{out} = \frac{I_{out}}{G_{coherent}} \tag{15}$$

From Eqn. (12) we find

$$G_{coherent} = \frac{1}{2\eta_o} \tag{16}$$

For an incoherent system the corresponding relationship is

$$I_{in} = \kappa \sum_{j}^{\sqrt{N}} \eta_{ij}^2 |E_{j\ out}|^2 = \sqrt{N}\eta_1^2 I_{out} = \frac{I_{out}}{G_{incoherent}} \tag{17}$$

In this case Eqn. (12) yields

$$G_{incoherent} = \frac{\sqrt{N}}{2\eta_o} \tag{18}$$

Note that $\frac{1}{G}$ is the total diffraction efficiency of the volume hologram. Since this must be less than 1 we know that $G > 1$. η_o is determined by the physical properties of the crystal, including the maximum charge density available for grating storage, the thickness of the crystal, and its the electrooptic coefficients. For small η_1 we may estimate η_o as $\eta_o \approx \Delta\epsilon \frac{2\pi}{\lambda} L$ where L is the length of the crystal along the optical axis. For $\Delta\epsilon \approx 10^{-5}$, $\lambda \approx 10^{-6}$ m and $L \approx 10^{-2}$ m, $\eta_o = o(1)$. This means that in coherent systems relatively little gain (i.e. $G = o(1)$) is needed to recall a large number of sinusoidal gratings stored in a photorefractive crystal. Of course as we attempt to store arbitrarily many gratings other limits arise, but at least over a finite bandwidth of the electrooptic response of the crystal coherent systems should have no difficulty in achieving interconnection densities on the order of those implied by the geometrical constraints. Incoherent systems, on the other hand, are unable to take advantage of holographic phase

matching and are thus less efficient [30]. In order to achieve $N = 10^{10}$, for example, we must supply a gain of $G = 10^5$ in each neural plane. Examples of how G may be obtained optically include various combinations of image intensifiers and spatial light modulators and multiwave mixing in nonlinear materials. For example, an optically addressed spatial light modulator such as the Hughes liquid crystal light valve is sensitive to approximately $10\mu w/cm^2$. If the read-out beam has an intensity of $1w/cm^2$ then we realize a gain of 10^5.

The choice between coherent and incoherent implementations of optical neural networks offers advantages and disadvantages on both sides. The incoherent system is easier to implement but requires the large gain described above and offers only unipolar activities and interconnection strengths. The coherent implementation offers bipolar activities and interconnections but requires rigid phase stability in the optical system over potentially very long learning cycles. This stability is not difficult to achieve in prescribed learning architectures, but may be more difficult to achieve in adaptive systems. In addition, coherent systems generally square the signal incident on the nonlinearity, unless interferometric detection is used. Interferometric detection is difficult to implement in a complex optical system. Although the incoherent system is straight forward to implement, this simplicity comes at a cost of requiring biasing to compensate for unipolar values and external gain. The coherent system is more elegant in that these additional mechanisms are not necessary, but it is more sensitive to specific design issues. One way of making coherent implementations more robust might be to include adaptive optics, such as phase conjugate devices, to compensate for phase instabilities. Although these devices might also be needed in adaptive incoherent systems to detect the phase of a grating in order to correctly update the associated interconnection, in the incoherent case it is only necessary to detect the current state of the phase. In the coherent case it is generally necessary to continuously track the phase.

4.4 Learning architectures

We now turn to the question of how we can specify the strength of each interconnection. There is a nice compatibility between simple (multiplicative)

Hebbian learning and holography; the strength of the connection between two neurons can be modified by recording a hologram with light from the two neurons. It is not possible, however, to record multiple holograms in a single crystal independently. Thus far we have shown that the space charge in a photorefractive crystal may be arranged to achieve a very large number of independent interconnections. The task that remains is to find a means of using optical beams from outside the crystal to correctly arrange the three dimensional charge distribution. In particular, we must find means to address the full three dimensional bandwidth of the crystal from two dimensional neural planes. In order to successfully implement learning with photorefractive crystals the nonlinear dynamics that govern the multiple exposure of holograms in a photorefractive medium must be reconciled with the nonlinear equations that describe the iterative procedures of learning algorithms. It is extremely difficult to fully characterize analytically the ability of an optical system to simulate a particular learning algorithm. We will have to rely heavily on experiment in the search for the optimum match between nonlinear optics and learning procedures for neural networks. In this section we describe learning architectures which are relatively simple to implement experimentally and which can be used to evaluate the capability of photorefractive crystals to store information in the form of connectivity patterns in a neural computer.

The first learning algorithm we consider is the prescribed sum of outer products of Eqn. (2). As we saw in the previous section, a sum of this sort may be implemented as a series of exposures of a volume hologram. In a photorefractive crystal, the exposure of a new hologram partially erases previously recorded holograms. This places an upper limit on the maximum number of holograms that can be recorded and thus the number of associations, M, that can be stored in the crystal. The limit is found by determining the minimum tolerable diffraction efficiency for each association and solving for the number of exposures that will yield this efficiency. Let A_m be the amplitude of the m^{th} hologram recorded. After a total of M exposures

$$A_m = A_o(1 - e^{\frac{-t_m}{\tau_r}}) \exp(-\sum_{m'=m+1}^{M} \frac{t_{m'}}{\tau_e}) \tag{19}$$

where A_o is the saturation amplitude of a hologram recorded in the pho-

torefractive crystal, t_m is the exposure time for the m^{th} hologram, τ_r and τ_e are, respectively, the characteristic time constants for recording and erasing a hologram in the crystal. We allow for the case that $\tau_e \neq \tau_r$ in light of limited evidence that this may be the case in some crystals [31]. Ionic conductivity is one mechanism leading to multiple time constants. We can use several different criteria for selecting the exposure schedule t_m. For instance if we require $A_m = A_{m+1}$ for all m we obtain

$$(1 - e^{\frac{-t_m}{\tau_r}})e^{\frac{-t_{m+1}}{\tau_e}} = (1 - e^{\frac{-t_{m+1}}{\tau_r}}) \tag{20}$$

If $\tau_r = \tau_e$ then the solution to Eqn. (20) under the boundary condition $t_1 \gg \tau_r$ is

$$t_m = \tau_e \ln\left(\frac{m}{m-1}\right) \quad m > 1 \tag{21}$$

which yields

$$A_m = A_M = \frac{A_o}{M} \tag{22}$$

For the case $\tau_r \neq \tau_e$ we define ρ_m such that $t_m = \rho_m \tau_e$. Since, from Eqn. (19), $\lim_{M\to\inf} A_1 = 0$ Eqn. (20) may be satisfied only if $\lim_{m\to\inf} t_m = 0$. Thus for some $m_o > 1$ $\rho_{m_o} \ll 1$ and $t_{m_o} \ll \tau_r$. Then, from Eqn. (20),

$$t_{m_o+1} \approx \frac{t_{m_o}}{1 + \frac{t_{m_o}}{\tau_e}} = \left(\frac{\rho_{m_o}}{1 + \rho_{m_o}}\right)\tau_e \tag{22}$$

or

$$\rho_{m_o+1} = \frac{\rho_{m_o}}{1 + \rho_{m_o}} \tag{23}$$

By induction, for $m > m_o$

$$\rho_m = \frac{1}{(m - m_o) + \frac{1}{\rho_{m_o}}} \tag{24}$$

As m grows large with m_o fixed, Eqn. (24) can be shown to yield

$$\rho_m \approx \frac{1}{m} \tag{25}$$

and

$$t_m = \frac{\tau_e}{m} \tag{26}$$

The value of m for which this approximation holds increases with the ratio $\frac{\tau_e}{\tau_r}$. In the case $\tau_r = \tau_e$ for example, $\frac{\tau_e}{3t_3} = 0.82$ and $\frac{\tau_e}{10t_{10}} = 0.95$. In any case, for $M \gg m_o$ for some m_o satisfying the constraints preceding Eqn. (22),

$$A_m = A_M = A_o(1 - e^{\frac{-\tau_e}{M\tau_r}}) \tag{26}$$

for all m. Solving for M with $A_m \ll A_o$ we find a limit for M given by

$$M \approx \frac{\tau_e}{\tau_r}\frac{A_o}{A_m} \tag{27}$$

This result agrees well with what we might expect intuitively. The number of exposures allowed increases in proportion with the ratio $\frac{\tau_e}{\tau_r}$ (if we erase slowly we can store more holograms) and the ratio of the maximum possible and minimum detectable grating amplitudes.

The second architecture we will discuss is capable of implementing the backwards error propagation algorithm [23,24] in a multilayered network. The architecture, shown in Fig. 4, is a variation on a system we have described previously [6]. The system as shown has two layers but an arbitrary number of layers can be implemented as a straightforward extension. An input training pattern is placed at plane N_1. The pattern is then interconnected to the intermediate (hidden) layer N_2 via the volume hologram H_1. A two dimensional spatial light modulator placed at N_2 performs a soft thresholding operation on the light incident on it, simulating the action of a 2-D array of neurons, and relays the light to the next stage. Hologram H_2 interconnects N_2 to the output plane N_4 where a spatial light modulator performs the final thresholding and produces a 2-D pattern representing the response of the network to the particular input pattern. This output pattern is compared to the desired output and the appropriate error image is generated (either optically or with the aid of an image detector and re-recording) on the spatial light modulator N_4. The undiffracted beams from N_1 and N_2 are recorded on spatial light modulators at N_3 and N_5, respectively. The signals stored at N_3, N_4, and N_5 are then illuminated from the right so that light propagates back towards the left. The back propagation algorithm demands a change in the interconnection matrix stored in H_2 given by

$$\Delta w_{ij}^{(2)} = -\alpha\varepsilon_i f'(x_i^{in})x_j^{out} \tag{28}$$

where α is a constant, ε_i is the error signal at the i^{th} neuron in N_4, x_i^{in} is the input diffracted onto the i^{th} neuron in N_4 from N_2, $f'(x)$ is the derivative of the thresholding function $f(x)$ which operates on the input to each neuron in the forward pass, and x_j^{out} is the output of the j^{th} neuron in N_2. Each neuron in N_4 is illuminated from the right by the error signal ε_i and the backward transmittance of each neuron is proportional to the derivative of the forward output evaluated at the level of the forward propagating signal. As we have described above, the hologram recorded in H_2 is the outer product of the activity patterns incident from N_4 and N_5. Thus the change made in the holographic interconnections stored in H_2 is proportional to the change described by Eqn. (28).

The change in the interconnection matrix stored in H_1 required under the back propagation algorithm is

$$\Delta w_{lm}^{(1)} = -\sum_i \alpha \varepsilon_i f'(x_i^{in}) w_{il}^{(2)} f'(x_l^{in}) x_m^o \tag{29}$$

where x_m^o is the activity on m^{th} input on N_1. The error signal applied to N_4 produces a diffracted signal at the l^{th} neuron in N_2 which is proportional to

$$-\sum_i \varepsilon_i f'(x_i^{in}) w_{il}^{(2)} \tag{30}$$

We assume that during the correction cycle for H_1 N_5 is inactive. Once again, if the backward transmittance of the l^{th} neuron is proportional to $f'(x_l^{in})$ then the change made to the hologram by the signals propagating back from N_2 and N_3 is proportional to the change prescribed in Eqn. (29).

A key element in this architecture is the assumption that the spatial light modulators at N_2 and N_4 may have transmittances which may be switched between a function $\frac{f(x)}{x}$ for the forward propagating signal and $f'(x)$ for the back propagating signal. In both cases x represents the forward propagating signal. Two of us (Wagner and Psaltis) have previously described how nonlinear etalon switches might be used in this application [7,8]. Electrooptic spatial light modulators might also be used [8].

We have performed an experiment to show how a single layer of error driven learning might be implemented. This experiment is shown schematically in Fig. 5. In this case, the stored vectors $\underline{x}^{(m)}$ correspond to two

dimensional patterns recorded on a liquid crystal light valve from a video monitor. The output vectors $\underline{y}^{(m)}$ correspond to the single bit output of the detector, D. The input vectors are imaged onto a photorefractive crystal via two separate paths. The strength of the grating between the image of the input along one path and the image along the other path is read out by light polarized orthogonally to the write beams. One of the write beams is circularly polarized while the other is linearly polarized. The polarizer, P, blocks the out of plane component of both the linearly polarized beam and the diffracted circularly polarized beam, passing only the in plane diffracted beam. This allows readout of the grating as it is recorded. The diffracted light is imaged onto the detector, D. This system is to classifies input patterns presented to it into two classes according to whether the output of the detector when the pattern is presented is high or low. If during training a pattern we would like to classify as high yields a low response then the hologram is reinforced by exposing the crystal to the interference of the two beams, each carrying the image of that pattern. This exposure continues until the diffracted output increases by a fixed amount. If a pattern which should be classified as low is found during training to yield a diffracted output that is too high then the hologram diffracting that pattern is erased by a fixed amount by exposing the crystal with only one of the imaging beams. (One beam is blocked by the shutter, SH). An experimental learning curve showing the diffracted intensities for each learning cycle for four training patterns in a system implemented using an Fe doped $LiNbO_3$ crystal is shown in Fig. 6. The system classifies the patterns 0 and 2 as high and 1 and 3 as low. At first all patterns are low. The first two learning cycles are intended to drive the outputs of 0 and 2 above threshold. However, they have the undesired effect of also driving pattern 3 above threshold. Thus in the third learning cycle 3 is erased. In this particular erase cycle the erasure was too severe. Notice that pattern 2 is erased in this cycle, even though there is no overlap between this pattern and pattern 3. The reason for this is that the two images of pattern 3 are in focus only over a limited region of the crystal volume. Outside of this region the unfocused image may erase the hologram formed by pattern 2. In the subsequent two cycles patterns 0 and 2 are again reinforced. This has the unwanted effect of driving both patterns 1 and 3 just above threshold. In the final two

cycles patterns 1 and 3 are erased until both are below threshold. At this point all patterns are correctly classified and learning stops.

In this experiment the photorefractive crystal acts as a two dimensional modulator. The diffraction efficiency between the two imaging paths is high where the patterns 0 and 2 overlap and low where patterns 3 and 1 overlap. As mentioned above, a problem arises in the fact that the overlap is well defined only in the image plane, meaning the crystal must be thinner than the depth of focus of the images. In order to utilize the full capacity of photorefractive volume holograms it will be necessary to move beyond this implementation to architectures utilizing the full three dimensional capacity of the crystal as discussed above. Nevertheless, this experiment demonstrates in a rudimentary way how learning in photorefractive crystals may proceed.

4.5 Conclusion

Photorefractive crystals represent a promising interconnection technology for optical neural computers. The ease of dynamic holographic modification of interconnections in these crystals allows the implementation of a large class of outer product learning networks. The density of interconnections which may be implemented in these crystals is limited by physical and geometrical constraints to the range of 10^8 to 10^{10} per cm^3. In order to achieve these limits consideration must be given to the exposure schedule of the crystal.

References

[1] Y. S. Abu-Mostafa and D. Psaltis, Optical Neural Computers, Scientific American, **256(3)**,88(1987).

[2] D. Psaltis, J. Yu, X. G. Gu, and H. Lee, Second Topical Meeting on Optical Computing, Incline Village, Nevada, March 16-18,1987.

[3] D. Psaltis, X. G. Gu, H. Lee, and J. Yu, Optical Interconnections Implemented with Volume Holograms, to be published.

[4] P. J. Van Heerden, Theory of optical information storage in solids, Appl. Opt., **2**,(4),393(1963).

[5] M. Cohen, Design of a new medium for volume holographic information processing, Appl. Opt., **25**(14),2288(1986).

[6] K. Wagner and D. Psaltis, Multilayer optical learning networks, Proc. SPIE 752-16,(1987).

[7] K. Wagner and D. Psaltis, Nonlinear etalons in adaptive optical neural computers, IEEE First Annual International Conference on Neural Networks, San Diego, California, June 21-24, 1987.

[8] K. Wagner and D. Psaltis, Multilayer optical learning networks, to appear in Appl. Opt.

[9] D. Z. Anderson, Adaptable interconnects for optical neuromorphs: demonstration of a photorefractive projection operator, Proceedings of the International Conference on Neural Networks, San Diego, June 1987.

[10] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, (1984).

[11] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. USA, **79**,2554(1982).

[12] S. S. Venkatesh and D. Psaltis, Information storage and retrieval in two associative nets, Conf. on neural network models for computing, Santa Barbara, California, April 1985.

[13] L. Personnaz, I. Guyon, and G. Dreyfus, Information storage and retrieval in spin-glass like neural networks, J. Physique Lett., **46**, L359(1985).

[14] D. Psaltis and C. Park, Nonlinear discriminant functions and associative memories, *Neural Networks for Computing*, APS conference proc. No. 151 (1986).

[15] T. Maxwell, C. L. Giles, Y. C. Lee and H. H. Chen, Nonlinear dynamics of artificial neural systems, *Neural Networks for Computing*, APS conference proc. No. 151 (1986).

[16] E. B. Baum, On the capabilities of multilayer perceptrons, to be published.

[17] D. Psaltis and N. H. Farhat, Optical information processing based on an associative memory model of neural nets with thresholding and feedback, Opt. Lett., **10**,(2), 98(1985).

[18] Y. Owechko, G. J. Dunning, E. Marom, and B. H. Soffer, Holographic associative memory with nonlinearities in the correlation domain, Appl. Opt. **26**,(10),1900(1987).

[19] B. Kosko and C. Guest, Optical bidirectional associative memories, SPIE Proc. **758**, Jan. 1987.

[20] R. A. Athale, H. H. Szu, and C. B. Friedlander, Optical implementation of associative memory with controlled nonlinearity in the correlation domain, **11**,(7),482(1986).

[21] F. Rosenblatt, *Principles of Neurodynamics: Perceptron and the Theory of Brain Mechanisms*, Spartan Books, Washington,(1961).

[22] B. Widrow and M. E. Hoff, Adaptive switching circuits, IRE Wescon Conv. Rec., pt. 4,96(1960).

[23] D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel Distributed Processing, Volume 1*, MIT Press, Cambridge, (1986).

[24] D. B. Parker, Learning logic, Invention report, S81-64, File 1, office of Technology Licensing, Stanford University, October 1982.

[25] J. S. Denker (Ed.), *Neural Networks for Computing*, APS conference proc. No. 151 (1986).

[26] A. D. Fisher, R. C. Fukuda, and J. N. Lee, Implementations of adaptive associative optical computing elements, Proc. SPIE **625**, 196(1986).

[27] K. Fukushima, A hierarchical neural network model for associative memory, Biol. Cybern. **50**,105(1984).

[28] S. Grossberg, *Studies of Mind and Brain*, D. Reidel Pub. Co., Boston, (1982).

[29] N. V. Kuktarev, V. B. Markov, S. G. Odulov, M. S. Soskin, and V. L. Vinetskii, Ferroelectrics, **22**,949(1979).

[30] J. W. Goodman, Fan-in and fan-out with optical interconnections, Optica Acta, **32**,(12),1489(1985).

[31] D. L. Staebler and W. Phillips, Fe-doped $LiNbO_3$ for read-write applications, Appl. Opt., **13**,(4),788(1974).

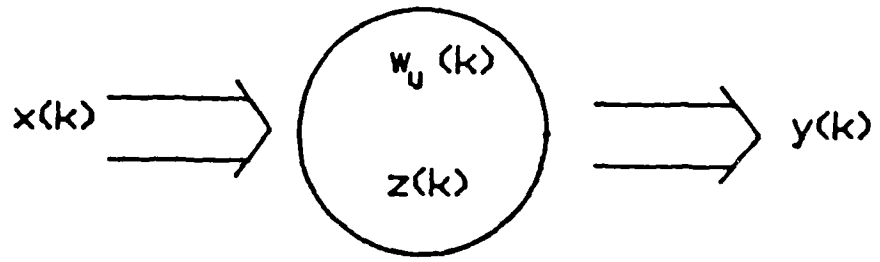


Figure 1. General neural network architecture.

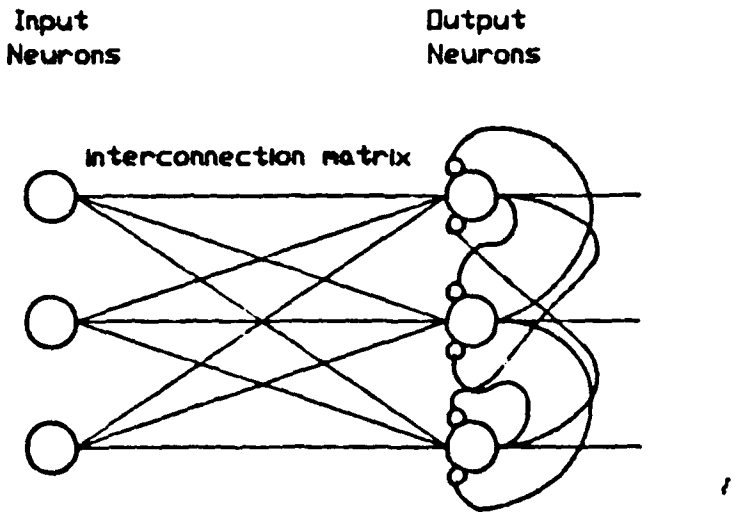


Figure 2. Two layer network with lateral inhibition. Connections ending with an open circle are inhibitory.

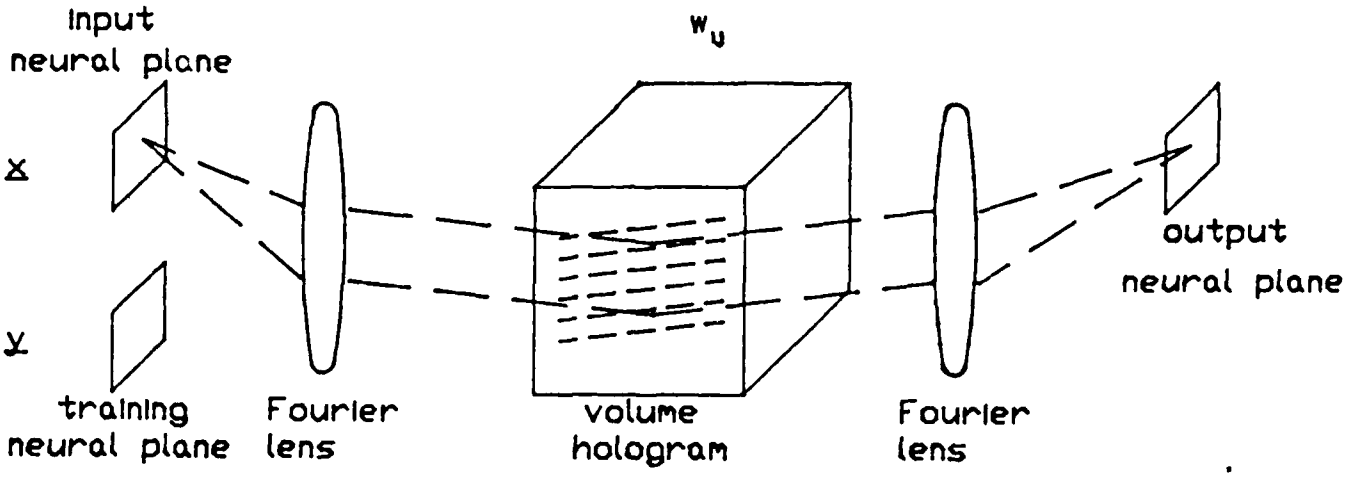


Figure 3. Optical neural computer architecture.

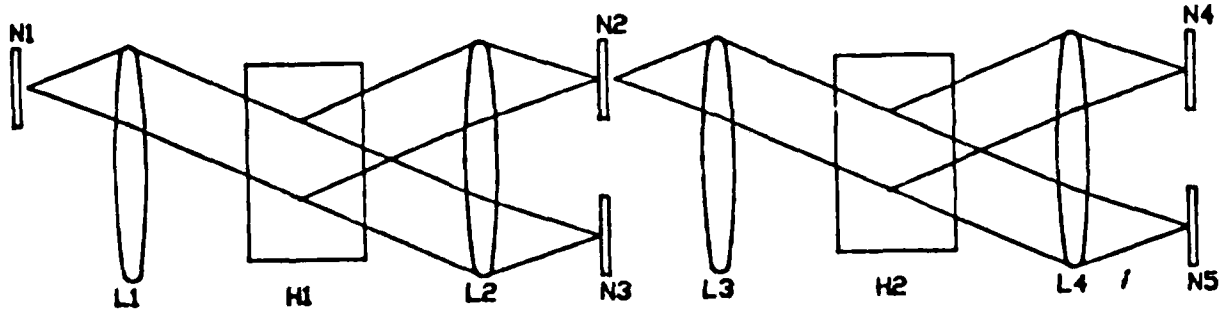


Figure 4. Optical architecture for backward error propagation learning.

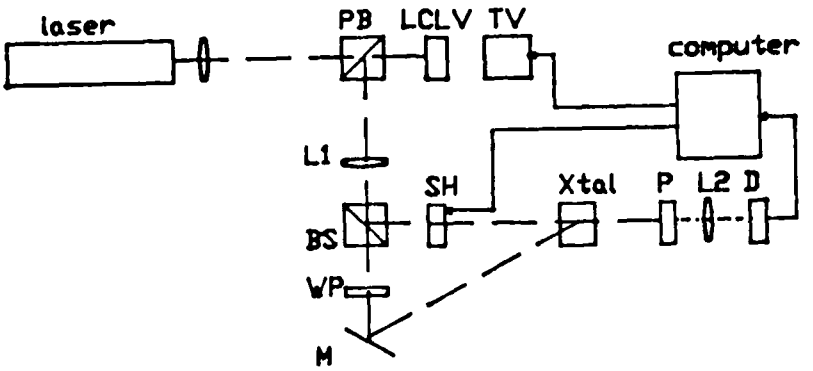


Figure 5. Simple photorefractive learning system. PB is a polarizing beamsplitter. L1 and L2 are imaging lenses. WP is a quarter waveplate. SH is a shutter. P is a polarizer. D is a detector. M is a mirror.

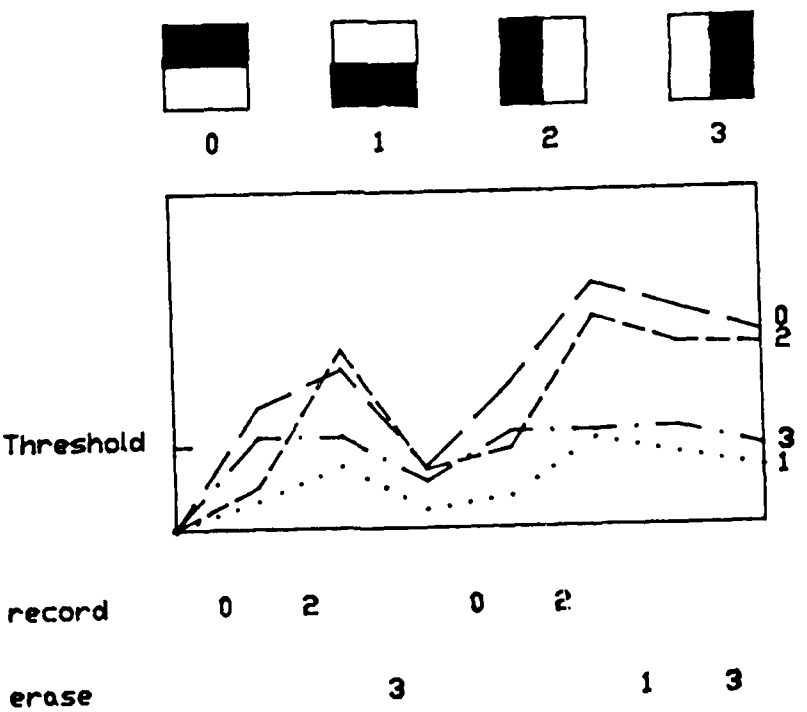


Figure 6. Experimental learning curves.

END

DATE

FILMED

6-88

DTIC